HTTP, CRUD & REST

ELEVATION ACADEMY

What are HTTP, CRUD and REST

These 3 acronyms make the WWW tick. How?

- They they describe and define how clients and servers should communicate.
- Their rules, guidelines and methods keep the web ordered and efficient.
- They are universally accepted and allow developers from all around the world to work together.

In order to make good APIs we need to understand REST, CRUD and HTTP!

- CRUD - the basic functions that can be used on data
- HTTP - the protocols that govern the web
- REST - an abstract architectural style

ELEVATION ACADEMY

HTTP methods & CRUD

POST	Create: add new data
GET	Read: view data
PUT	Update: change data
DELETE	Delete: remove data

HTTP Status Codes

Status codes indicate the result of the HTTP request.

1XX - informational
2XX - success
3XX - redirection
4XX - client error
5XX - server error

HTTP Header

A set of fields that operating parameters of an HTTP request or response. For example:

Field Name	Description / Function
Authorization (req)	Authentication credentials for HTTP authentication
Referer (req)	The address of the previous web page from which a link to the currently requested page was followed.
Content-Length (res)	The length of the response body in octets (8-bit bytes)
Expires (res)	The date/time after which the response is considered stale

HTTP Media Types

`Content-Type` A header field used by servers to inform client of the data type of the response; or by clients making POST or PUT requests.

`application/xml`	Tells the client that the data being sent in the request is XML.
`application/x-www-form-urlencoded`	Tells the server that the client is sending encoded form data.

`Accept`

A header field used by the client to request the data type it wants as a response

`application/json`	Client requesting a response in JSON.
`text/html`	Client requesting a response in HTML.

ELEVATION ACADEMY

REST (Representational State Transfer)

an architectural style for designing distributed systems eg The Internet.

REST-ful systems should be:

- Client and Server - separation of view and data
- Stateless - nether client or server should care about the state of the other
- Layered - client cannot tell whether it is connected directly to the end server, or to another server along the way
- Cacheable - server responses may be cached to improve performance
- Uniformly Interfaced: Resources are identified by Uniform Resource Identifiers (URI) and HTTP methods allow them to be manipulated.

REST (Representational State Transfer)

Resource: "the thing that is accessed by the URL you supply" eg

`/teachers (URL)` => **all the teachers (resource)**
`/teachers/1` => **the teacher with ID 1**
`/customers/21/reviews` => **the reviews of the customer 21**

A **representation** of a resource is transferred between hosts. This can be in any format eg JSON XML

```
{
  "id":1,
  "name":"Steven",
  "profession":"Teacher"
}
```


HTTP methods / URIs for collection/item

	http://api.co/v2/cars/	http://api.co/v2/cars/1234
GET	List all the cars	Retrieve an individual car
POST	Create a new car	Error
PUT	Replace the entire collection with a whole new list of cars	Replace or create an individual car
DELETE	Delete all the cars	Delete an individual car

Safe HTTP Methods

Safe methods don’t modify data.

A GET request is safe because it doesn't modify the resource you are requesting.

POST, PUT and DELETE are not safe!

Idempotent HTTP Methods

An idempotent request can be made times and always produces the same result.

GET, DELETE, and PUT should be idempotent

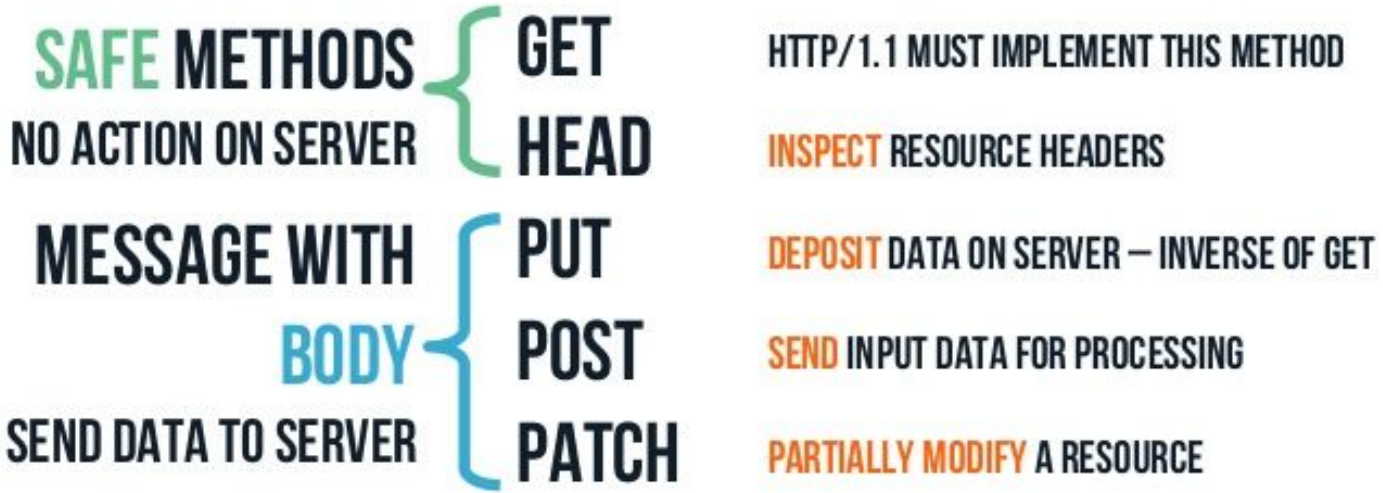
POST are not idempotent

HTTP Message Body

The data transmitted following the headers.

All responses should have a body (apart from those to HEAD requests).

But not all requests should have bodies!!!



HTTP Method Summary

HTTP Method	RFC	Request Has Body	Response Has Body	Safe	Idempotent	Cacheable
GET *	RFC 7231	No	Yes	Yes	Yes	Yes
HEAD	RFC 7231	No	No	Yes	Yes	Yes
POST *	RFC 7231	Yes	Yes	No	No	Yes
PUT *	RFC 7231	Yes	Yes	No	Yes	No
DELETE *	RFC 7231	No	Yes	No	Yes	No
CONNECT	RFC 7231	Yes	Yes	No	No	No
OPTIONS	RFC 7231	Optional	Yes	Yes	Yes	No
TRACE	RFC 7231	No	Yes	Yes	Yes	No
PATCH	RFC 5789	Yes	Yes	No	No	Yes

*common methods

Node.js & Express

ELEVATION ACADEMY

What is Node.js

Allows us to execute JS outside of the browser environment.

Therefore we can develop servers with Node.js!

A Node server can respond to clients by giving them the files or data that was requested.

A Node server can also communicate with a Database server in order to perform CRUD operations.

How to run?

Node code can be executed in shell or via a JS file.

DEMO TIME!

ELEVATION ACADEMY

module

A single file that encapsulates related code into a unit of code. An NPM package is one or more module packaged together.

module.exports

An object belonging to a module that holds the properties that you wish other modules to access.

require

Used to import the object that was exported from another file.

Building a Server with Express

Express is a framework that makes it easy to develop servers with Node.js

DEMO TIME!

ELEVATION ACADEMY

Looking forward...

- All projects will have client code and server code
- No more opening html files directly in the browser
- Whilst developing our server and client code will be executing on the same machine (soon to be joined by a Database server)
- In reality remember a client can be on any user PC & server is normally remote or cloud-based