# REST API

What it means to be RESTful

# What is REST

- REST stands for REpresentational State Transfer

- It is neither a standard nor a protocol

- REST is not HTTP, although because of HTTP's prevalence and popularity the two tend to go hand-in-hand

# What is REST

**In practice, REST is a flexible set of guidelines to allow for the traversal and consumption of network-available information.**

# What is REST

- The general acceptance of REST is all in an effort to be able to easily and accurately access and interact with meaningful, conventional, and unique addresses pointing at certain representations of data

- What a user requests will be consistently served to the user in an intuitive manner and from that point on, after the initial request, a user can navigate and hop from state to state in order to arrive at the final combination of HTML, CSS, and JavaScript

# What is REST

- RESTful architecture is commonly associated with:
  - Well-defined URIs that “represent” some kind of resource, such as “/posts” on a blog representing posts on the blog
  - HTTP methods being used as verbs to perform actions on that resource (I.e. GET for reading posts, POST for writing posts)
  - The ability to access multiple format representations of the same data (I.e. both JSON and XML representations of a blog post)
  - Sending HTTP status codes for a more detailed response

# APIs

- API stands for Application Programming Interface.
- Can be defined as a set of functions and procedures allowing the creation of applications that access the features or data of an operating system, application, or other service.
- For our purposes, they will be endpoints that provide the ability to interact with, write data to, and/or receive data from a web application.

# APIs

- Interactions will be facilitated by HTTP verbs, such as:
    - GET - used to receive data from web application
    - PUT - used to update specific data from web application
    - POST - used to write new data to a web application
    - DELETE - used to delete data from a web application

# REST API

- Let’s say we are building a blog, and we want to represent users and blogs on our platform.

- If we want to allow access to see users, we could set up a route at ‘/api/users’ to serve up our users on a GET request.

- Similarly, we could serve up blog posts at ‘/api/blogs’ on a GET request.

# REST API

- To show a specific user, we could extend our users route to include “/api/users/:userId” on a GET request.

- Conversely, to update a user we could access the same route of ‘/api/users/:userId”, but access it this time via a PUT request, providing a payload of data that can be used to alter the user.

# REST API

- Let’s say, though, that we wanted to allow a specific post belonging to a specific user to be accessed or updated. How could we represent this?
- We can think of it as blog posts being a result of users, and as such could provide a route like the following to receive or update a blog post on a GET or PUT request respectively:

```
/api/users/:userId/posts/:postId
```

- By setting up our API this way, the navigation to a user’s specific blog feels more intuitive, as we will look at our users for a specific user, and then look for a blog post pertaining to them.

# REST API

- Having an intuitive API can help developers to better plan and organize their data, and can also inform how views will be routed on the front end as well.