# URL Paths

The URL Path comes right after the domain (or port if one is provided) in a URL string.

In this URL:

http://testdomain.com/root/next

The path is:

`/root/next`

## Path conventions

Many static websites (websites where the content does not change, as opposed to dynmaic web applications) use paths as a direct mapping to the path to the server's file system. For example, If I was running a static web server for "mystaticstate.com" from the root of my file system, a `GET` request to `http://mystaticstate.com/documents/hello.txt` would serve the file at `/documents/hello.txt` from my server.

Most dynamic web applications don't use this simple mapping of `URL path` -> `file path`. Technically, a URL path is just a string that the web server can do what it wants with, and modern websites take advantage of that flexibility. Some common examples of what paths are used for include:

- The hierarchy of pages on a website, whether or not that reflects a server's file structure
- Parameters passed into an HTTP request, like the ID of a resource
- The version of the API
- The type of resource being requested

## RESTful APIs

Representational State Transfer, or REST, is a popular convention that many dynamic HTTP servers follow. Not all HTTP APIs are "REST APIs", or "RESTful", but it is very common.

RESTful servers follow a loose set of rules that makes it easy to build reliable and predictable web APIs. REST is a set of conventions about how HTTP APIs should be built.

### Separate and agnostic

The big idea behind REST is that resources are transferred via well-recognized, language-agnostic client/server interactions. A RESTful style means the implementation of the client and server can be created independently of one another, as long as some simple standards, like the names of the available resources, have been established.

### Stateless

A RESTful architecture is stateless, which means the server does not need to know what state the client is in, nor does the client need to know what state the server is in. Statelessness in REST is enforced by interacting with resources instead of commands. Keep in mind, this doesn't mean the applications are stateless - what would "updating a resource" even mean if the server wasn't keeping track of its state?

### Paths in REST

In a RESTful API, the last section of the `path` of a URL specifies the resource. In Jello, this means an `issue`, a `user`, or a `project`. Depending on whether the request is a `GET`, `POST`, `PUT` or `DELETE`, the resource is read, created, updated, or deleted.

The Jello API we have been working with is a RESTful API! Take a look at the URLs we've been using:

- `https://api.boot.dev/v1/courses_rest_api/learn-http/projects`
- `https://api.boot.dev/v1/courses_rest_api/learn-http/users`
- `https://api.boot.dev/v1/courses_rest_api/learn-http/issues`

1. The first part of the path specifies the version. In this case, version 1, or `v1`.
2. The second part of the path tells our server that this is a REST API for the "Learn HTTP" course.
3. The last part denotes which resource is being accessed, be it a `project`, `user`, or `issue`.

## URL Query Parameters

A URL's query parameters appear next in the URL structure but are not always present - they're optional. For example:

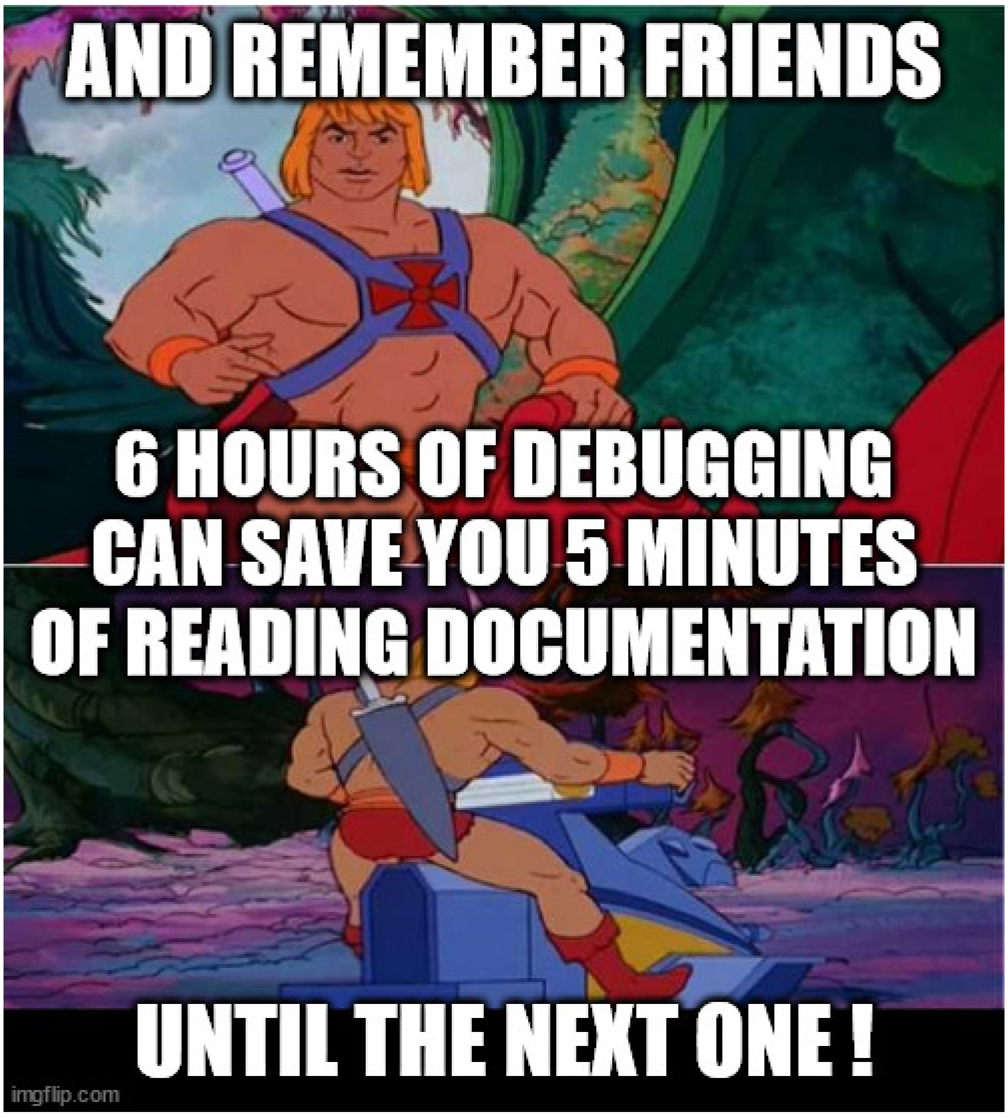
https://www.google.com/search?q=boot.dev

`q=boot.dev` is a query parameter. Like headers, query parameters are `key / value` pairs. In this case, `q` is the key and `boot.dev` is the value.

## The Documentation of an HTTP Server

It's the responsibility of a server's developers to provide you with instructions, or documentation that explains how to interact with it. For example, the documentation should tell you:

- The domain of the server
- The resources you can interact with (HTTP paths)
- The supported query parameters
- The supported HTTP methods
- Anything else you'll need to know to work with the server


AND REMEMBER FRIENDS
6 HOURS OF DEBUGGING CAN SAVE YOU 5 MINUTES OF READING DOCUMENTATION
UNTIL THE NEXT ONE !
imgflip.com

The server is the captain

The server has complete control over how the path in a URL is interpreted and used in a request. The same goes for query parameters. While there are a lot of strong conventions around how servers should interpret paths and query parameters, the server can do whatever it wants. That's why you need docs.

## Multiple Query Parameters

Query parameters are `key/value` pairs - that means there can be multiple pairs!

`http://example.com?firstName=lane&lastName=wagner`

In the example above:

- `firstName=lane`
- `lastName=wagner`

The `?` separates the query parameters from the rest of the URL. The `&` is then used to separate each additional pair of query parameters after that.