Uniform Resource Identifiers

We briefly touched on URLs earlier, let's dive a little deeper into the subject.

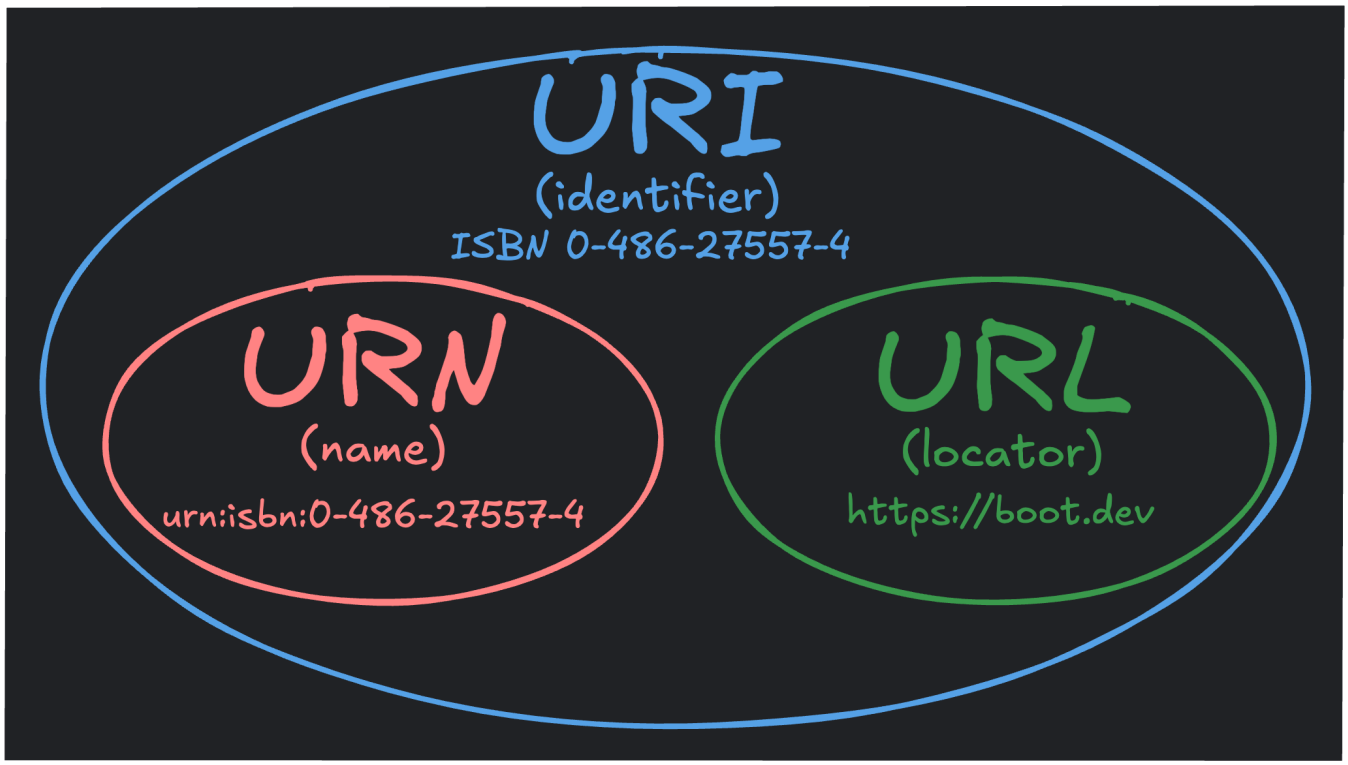
A URI, or Uniform Resource Identifier, is a unique character sequence that identifies a resource that is (almost always) accessed via the internet.

Just like JavaScript has syntax rules, so do URIs. These rules help ensure uniformity so that any program can interpret the meaning of the URI in the same way.

URIs come in two main types:

URLs URNs

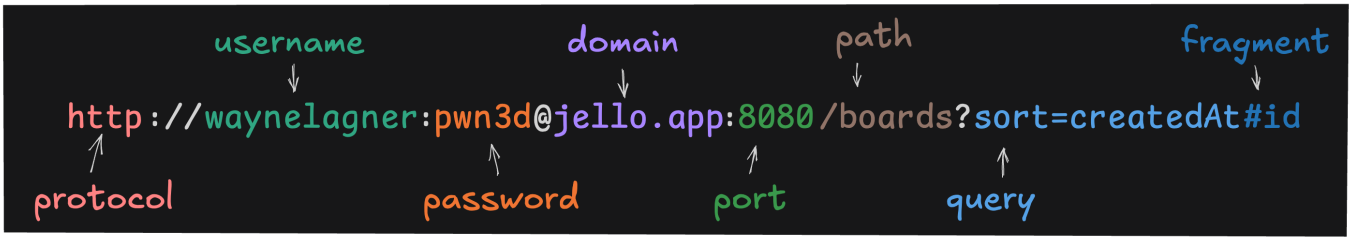
We will focus specifically on URLs in this course, but it's important to know that URLs are only one kind of URI.



Sections of a URL

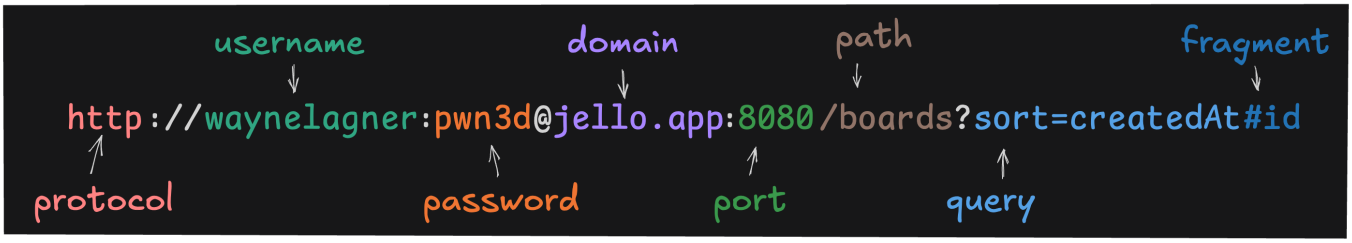
URLs have quite a few sections. Some are required, some are not.

1. **Scheme** (required): Indicates the protocol to be used (e.g., `http`, `https`, `ftp`).
2. **Host** (required): Specifies the domain name or IP address of the server.
3. **Port** (optional): An optional section that specifies the port number (e.g., `:80`, `:443`).
4. **Path** (required): The path to the resource on the server (e.g., `/path/to/resource`).
5. **Query** (optional): An optional section that contains data to be sent to the server (e.g., `?key1=value1&key2=value2`).
6. **Fragment** (optional): An optional section that points to a specific part of the resource (e.g., `#section1`).



URL Parts and Their Descriptions

There are 8 main parts to a URL, though not all the sections are always present. Each piece plays a role in helping clients locate the resources they're looking for.



Part	Required	Description
Protocol (Scheme)	Yes	Indicates the protocol to be used (e.g., `http`, `https`, `ftp`).
Username	No	Optional user information for authentication.
Password	No	Optional password for authentication.
Domain	Yes	Specifies the domain name or IP address of the server.
Port	No (defaults to 80 or 443)	Specifies the port number.
Path	No (defaults to /)	The path to the resource on the server.
Query	No	Contains data to be sent to the server.
Fragment	No	Points to a specific part of the resource.

The Protocol (Scheme)

The "protocol" (also referred to as the "scheme") is the first component of a URL. It defines the rules by which the data being communicated is displayed, encoded or formatted.

Some examples of different URL protocols:

- http
- ftp
- mailto
- https

For example:

- `http://example.com`
- `mailto:noreply@jello.app`

Not all schemes require a "//"

The "http" in a URL is always followed by `://`. All URLs have the colon, but the `//` part is only included for schemes that have an authority component. As you can see above, the `mailto` scheme doesn't use an authority component, so it doesn't need the slashes.

The **authority component** of a URL typically includes the user information (optional), host (domain name or IP address), and port (optional). It is the part of the URL that specifies the authority governing the namespace of the URL.

To have authority in the context of a URL means that a specific entity (typically a server or domain) is responsible for hosting, managing, or governing access to the resource described by the URL.

For example, in `http://username:password@example.com:8080/path`, the authority component is `username:password@example.com:8080`.

URL Ports

The port in a URL is a virtual point where network connections are made. Ports are managed by a computer's operating system and are numbered from `0` to `65,535` (Though port `0` is reserved for the system API).

Whenever you connect to another computer over a network, you're connecting to a specific port on that computer, which is listened to by a program on that computer. A port can only be used by one program at a time, which is why there are so many possible ports.

The port component of a URL is often not visible when browsing normal sites on the internet, because 99% of the time you're using the default ports for the HTTP and HTTPS schemes: `80` and `443` respectively.

Whenever you aren't using a default port, you need to specify it in the URL. For example, port `8080` is often used by web developers when they're running their server in "test mode" on their own machines.

Common Port Numbers and Their Uses

Here are some common port numbers and their typical uses:

- **20, 21**: FTP (File Transfer Protocol)
- **22**: SSH (Secure Shell)
- **25**: SMTP (Simple Mail Transfer Protocol)
- **53**: DNS (Domain Name System)
- **80**: HTTP (Hypertext Transfer Protocol)
- **110**: POP3 (Post Office Protocol)
- **143**: IMAP (Internet Message Access Protocol)
- **443**: HTTPS (HTTP Secure)
- **3306**: MySQL Database
- **5432**: PostgreSQL Database
- **6379**: Redis

Knowing these common ports can help you understand network traffic and troubleshoot connectivity issues.

Selecting an Empty Port

If you need an empty port, you can use any port number that is not already in use by another service on your machine. Ports in the range of **49152 to 65535** are designated as dynamic or private ports and are less likely to be used by standard services. You can use tools like `netstat` or `lsof` to check which ports are currently in use on your system.

URL Paths

On static sites (like blogs or documentation sites) a URL's path mirrors the server's filesystem hierarchy.

For example, if the website `https://exampleblog.com` had a static web server running in its /home directory, then a request to `https://exampleblog.com/site/index.html` would probably return the file located at `/home/site/index.html`.

But technically, this is just a convention. The server could be configured to return any file or data given that path.

It's not always that simple

Paths in URLs are essentially just another type of parameter that can be passed to the server when making a request. For dynamic sites and web applications, the path is often used to denote a specific resource or endpoint.

Query Parameters

Query parameters in a URL are not always present. In the context of websites, query parameters are often used for marketing analytics or for changing a variable on the web page. With website URLs, query parameters rarely change which page you're viewing, though they often will change the page's contents.

That said, query parameters can be used for anything the server chooses to use them for, just like the URL's path.

How Google uses query parameters

1. Open a new tab and go to `https://google.com`.
2. Search for the term "hello world"
3. Take a look at your current URL. It should start with `https://www.google.com/search?q=hello+world`
4. Change the URL to say `https://www.google.com/search?q=hello+universe`
5. Press "enter"

You should see new search results for the query "hello universe". Google chose to use query parameters to represent the value of your search query. It makes sense - each search result page is essentially the same page as far as HTML structure and CSS styling are concerned - it's just showing you different results based on the search query.