A Browser for Developers

In this lesson, we'll get an introduction to 'cURL', the browser for developers.

We'll cover the following

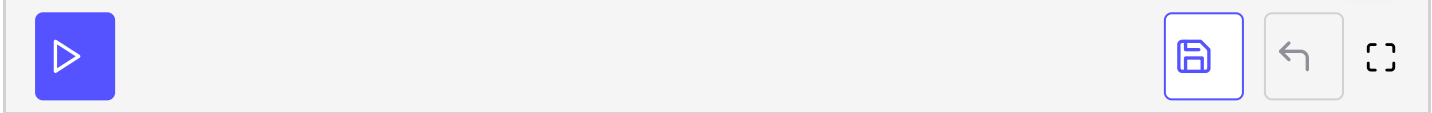
By now, we should understand the simple but important concept that browsers are simply HTTP clients built for the average web surfer.

They are more powerful than a platform's bare HTTP client (think of NodeJS's `require('http')`, for example), but at the end of the day, they're just a natural evolution of simpler HTTP clients.

Introducing `cURL`

As developers, our HTTP client of choice is usually cURL by Daniel Stenberg. It is one of the most popular software programs web developers use on a daily basis. It allows us to do an HTTP exchange on-the-fly by sending an HTTP request from our command line.

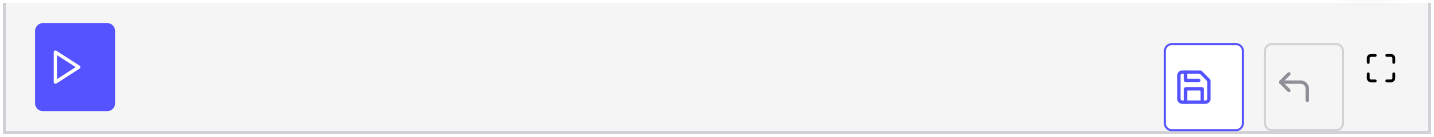
```
curl -I educative.io
```



In the example above, we have requested the document at `educative.io`, and Educative's server replied successfully. Rather than dump the response's body to the command line, we've used the `-I` flag to tell cURL we're only interested in the response headers.

Taking it one step further, we can instruct cURL to dump some more information, including the actual request it performs, so that we can have a better look at the entire HTTP exchange. The option we need to use is `-v` (verbose).

```
curl -I -v educative.io
```



The same information is available in mainstream browsers through their DevTools. As we've seen, browsers are nothing more than elaborate HTTP clients. Sure, they add an enormous number of features like credential management, bookmarking, and history but the truth is that they were born as HTTP clients for humans. This is important because in most cases you don't need a browser to test your web application's security, as you can simply curl it and take a look at the response.

One final thing I'd like us to understand is that anything can be a browser. If you have a mobile application that consumes APIs through the HTTP protocol, then the app is your browser. It just happens to be a highly customized one that you built yourself, one that only understands a specific type of HTTP responses from your own API.

Into the HTTP protocol

As we mentioned, the HTTP exchange and rendering phases are the ones that we're going to cover, as they provide the largest number of attack vectors for malicious users.

Take a quiz to test your knowledge of browsers in the next lesson!