So we've constructed an XHR object named `asyncRequestObject`. There are a number of methods that are available to us. One of the most important is the open method.

```
asyncRequestObject.open();
```

`.open()` takes a number of parameters, but the most important are its first two: the HTTP method URL to send the request

If we want to asynchronously request the homepage from the popular high-res image site, Unsplash, we'd use a `GET` request and provide the URL:

```
asyncRequestObject.open('GET', 'https://unsplash.com');
```

A little rusty on your HTTP methods?

The main two that you'll be using are:

- `GET` - to retrieve data
- `POST` - to send data

For more info, check out our course on HTTP & Web Servers!

> **Warning:** For security reasons, you can only make requests for assets and data on the same domain as the site that will end up loading the data. For example, to asynchronously request data from google.com your browser needs to be on google.com. This is known as the same-origin policy. This might seem extremely limiting, and it is!
>
> The reason for this is because JavaScript has control over so much information on the page. It has access to all cookies and can determine passwords since it can track what keys are pressed. However, the web wouldn't be what it is today if all information was bordered off in its own silos. The way to circumvent the same-origin policy is with CORS (Cross-Origin Resource Sharing). CORS must a technology that is implemented on the server. Services that provide APIs use CORS to allow developers to circumvent the same-origin policy and access their information.

QUESTION 1 OF 2

Go to Google, open up the developer tools, and run the following on the console:

```
const req = new XMLHttpRequest();
req.open('GET', 'https://www.google.com/');
```

What happens?

- The Google homepage open in the browser
- An async request sent to https://www.google.com
- Nothing happens
- An error occurs

SUBMIT

QUESTION 2 OF 2

```
const myAsyncRequest = new XMLHttpRequest();
myAsyncRequest.open('GET', 'https://udacity.com/', false);
```

- Nothing special, this is the standard way `.open()` works.
- The request is sent immediately.
- The JavaScript freezes and waits until the request is returned.

SUBMIT

NEXT