Async

ruby not meant to be aysnc

diagram:

the server/api gives data

the server has to look at a db

and is allowed 10 requests

but can queue many of the requests

Asynchronous

doesn’t need to wait for hte response

has a listener

check the diagram

send()

Synchronous ruby

does one request after the next

// is used instead of https => javascript quirk

hits the https otherwise http

github\_url = "https://api.github.com/events"

for html just type html in atom and tab complete

————————————————————————

the user sends a request using the http protocol using get post etc. the server will respond to with the header and payload and status

This should be done async so that the user does not have to wat therefore, the xhr.open(…, true) needs to be set

this XMLHttpRequest object allows the user to get the status of the server etc.

When async is being carried out the onreadystate is checked , onready has 5 states and we just need the one when it is done

The code will continue to run in the background

The processRequest will be called so that when this is true and the response is fulfilled it is carried out because of the eventListener which listens for when this becomes true

——————————————————————————

Our code just keeps running. What we need is a way to send our request and then be notified of when the request comes back so that our code can finish what it started.

To satisfy that need, that's why we have **events**.

More specifically for our case, that's why we have the readystatechange event that is fired by our XMLHttpRequest object whenever your request hits an important milestone on its epic journey.

all we need is some code to parse the result of the HTTP request inside our newly added event handler...

As it turns out, it isn't that simple. The complication comes from the readystatechange event being tied to our XMLHttpRequest object's readyState property. This readyState property chronicles the path your HTTP request takes, and each change in its value results in the readystatechange event getting fired. What exactly is our readyState property representing that results in its value changing so frequently? Check out the following table:

For every HTTP request that you make, your readyState property hits each of these five values. This means your readystatechange event gets fired five times. As a result, your processRequest event handler gets called five times. See where the problem is? Four out five times processRequest gets called, it won't be getting called for the reasons we are interested in - that is, the request has returned and its time to analyze the returned data.

 HTTP requests is status code **200**. This code is returned by the server when the HTTP request was successful.

the request made is a using get - it makes a request to the server ip info

if there is a response ie. no 404 and the readystate is ready then the response returned is okay. It receives the json as a string

this string is parsed using JSON.parse and returned as JSON which can then be queried i.e get the ip

What you can rely on is that the responseText property will return the raw string-based data for you to further process.

<https://www.kirupa.com/html5/making_http_requests_js.htm>