**Backend Notes**

**res.send vs res.json vs res.end** [more](https://medium.com/gist-for-js/use-of-res-json-vs-res-send-vs-res-end-in-express-b50688c0cddf#:~:text=But%20the%20main%20difference%20between%20res.json%20and%20res.send,uses%20json%20replacer%20and%20json%20spaces%20application%20settings.)

res.send

- sends response and then end the response

- it also set Content-Type according to data passed into res.send(), if body passed is buffer then Content-type will be application/octet-stream

res.json

* it is identical to res.send() when an array or object is passed, but also converts non-objects to json

res.end

* don’t return any response, it just ends the response
* it can be useful if no need to send any response like 404 page

**PUT VS PATCH**

* **PUT** is a method of modifying resource where the client sends data that updates the entire resource. It is used to set an entity’s information completely. PUT is similar to POST in that it can create resources, but it does so when there is a defined URI. **PUT overwrites the entire entity if it already exists, and creates a new resource if it doesn’t exist.**
* Unlike PUT, **PATCH applies a partial update to the resource**.
* This means that you are only required to send the data that you want to update, and it won’t affect or change anything else. So if you want to update the first name on a database, you will only be required to send the first parameter; the first name.

**req.query vs req.params**

* [req.params](http://expressjs.com/en/api.html#req.params) contains route parameters (in the path portion of the URL), and [req.query](http://expressjs.com/en/api.html#req.query) contains the URL query parameters (after the ? in the URL).

**Synchronous vs Asynchronous**

* synchronous is blocking functions which waits until execution completes
* Asynchronous is non-blocking functions which don’t wait for execution completion

**How to make synchronous?**

* async await
* callback
* const request = require(sync-request)

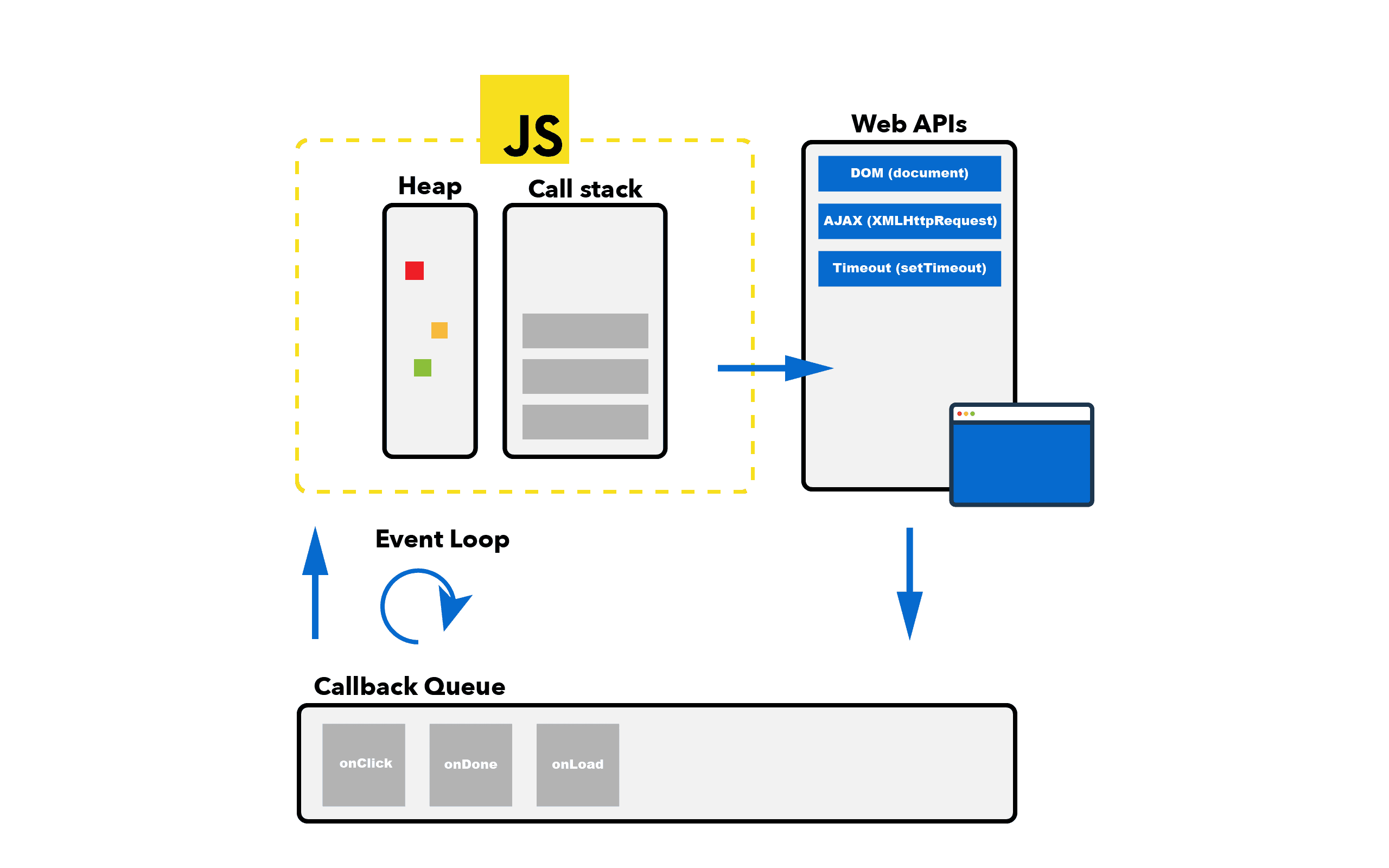
**Why is Node.js single threaded?**

* Initially, node.js was created as an experiment in asynchronous processing and in theory was that doing asynchronous processing on a single thread could provide more performance and scalability under typical web loads than the typical thread-based implementation when the application isn’t doing CPU intensive stuff and can run thousands more concurrent connections than Apache or IIS or other thread-based servers.
* There is also a very well known and criticized issue with the one thread per request model for a server which is that they don’t scale very well for several scenarios compared to the event loop thread model, in short, they lack scalability as the application grows to meet the future demands and with the addition of new features.

**Advantages of single threaded event loop**

* Can handle more & more concurrent client’s requests with ease.
* Eliminates the need of creating more and more threads, because of the Event loop.
* Applications built on top of node.js use the least threads possible to reduce memory or resource usage.

**Event loop or How it handles concurrency if it is single threaded?**

****

Header payload

Stringify

tostring vs tojson