JavaScript: Task1

1.Difference Between HTTP1.1 vs HTTP2

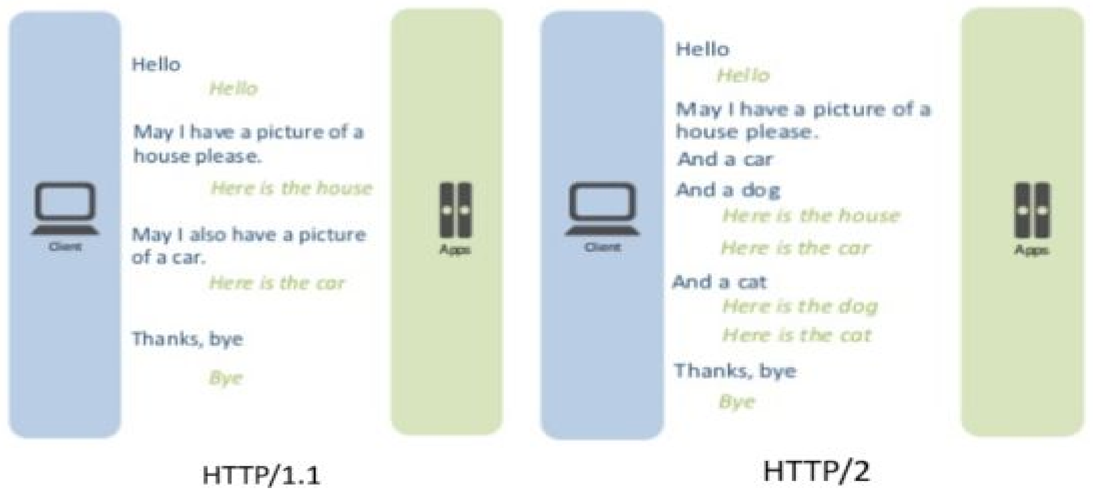
HTTP/1.1 was developed by Timothy Berners-Lee in 1989 as a communication standard for the World Wide Web.
HTTP/2 began as the SPDY protocol, developed primarily at Google with the intention of reducing web page load latency by using techniques such as compression, multiplexing, and prioritization.

HTTP/1.1, which keeps all requests and responses in plain text format.
HTTP/2 uses the binary framing layer to encapsulate all messages in binary format.

HTTP/1.1 loads resources one after the other, so if one resource cannot be loaded, it blocks all the other resources behind it. In contrast.
HTTP/2 is able to use a single TCP connection to send multiple streams of data at once so that no one resource blocks any other resource.

HTTP/1.1 is relatively secure since it uses digest authentication, NTLM authentication.
Security concerns from previous versions will continue to be seen in HTTP/2.

HTTP/1 Expands on the caching support by using additional headers like cache-control, conditional headers like If-Match and by using entity tags.
HTTP/2 does not change much in terms of caching. With the server push feature if the client finds the resources are already present in the cache, it can cancel the pushed stream.



2.HTTP VERSION HISTORY

HTTP/0.9 — The One-line Protocol

- Initial version of HTTP — a simple client-server, request-response, telenet-friendly protocol
- Request nature: single-line (method + path for requested document)
- Methods supported: `GET` only
- Response type: hypertext only
- Connection nature: terminated immediately after the response
- No HTTP headers (cannot transfer other content type files), No status/error codes, No URLs, No versioning

HTTP/1.0 — Building extensibility

- Browser-friendly protocol
- Provided header fields including rich metadata about both request and response (HTTP version number, status code, content type)
- Response: not limited to hypertext (`Content-Type` header provided ability to transmit files other than plain HTML files — e.g. scripts, stylesheets, media)
- Methods supported: `GET` , `HEAD` , `POST`
- Connection nature: terminated immediately after the response

Establishing a new connection for each request — major problem in both HTTP/0.9 and HTTP/1.0.

Both HTTP/0.9 and HTTP/1.0 are required to open up a new connection for each request (and close it immediately after the response was sent). Each time a new connection establishes, a TCP three-way handshake should also occur. For better performance, it was crucial to reduce these round-trips between client and server. HTTP/1.1 solved this with persistent connections.

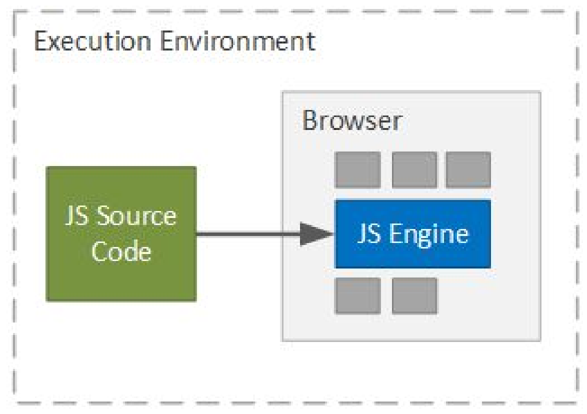
HTTP/1.1 — The standardized protocol

- This is the HTTP version currently in common use.
- Introduced critical performance optimizations and feature enhancements — persistent and pipelined connections, chunked transfers, compression/decompression, content negotiations, virtual hosting (a server with a single IP Address hosting multiple domains), faster response and great bandwidth savings by adding cache support.
- Methods supported: `GET` , `HEAD` , `POST` , `PUT` , `DELETE` , `TRACE` , `OPTIONS`
- Connection nature: long-lived

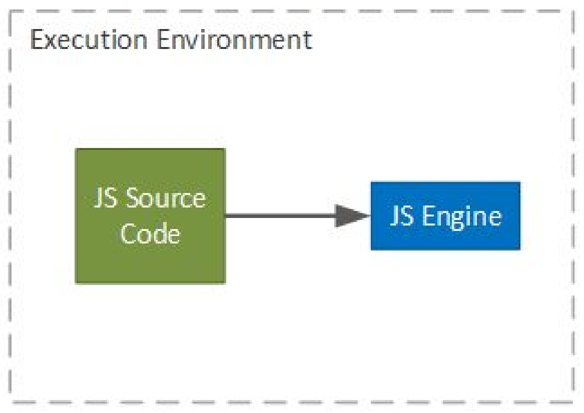
HTTP/2.0 and the future

All above features are being used by major web servers and browsers today. But modern enhancements like HTTP/2.0, Server Side Events (SSE), and Websockets have changed the way that the traditional HTTP works.

3.DIFFREENCES BETWEEN BROWSER JS AND NODE JS



JAVASCRIPT IN A BROWSER



STANDALONE JAVASCRIPT

Nodejs is a javascript runtime based on google chrome javascript engine v8.In simple words, we can say it's just the javascript engine v8 running standalone.

Browser.js is mainly used for client-side applications like validations on a web page or dynamic page display and as the name suggests it gets executed in the browser only.

Node.js javascript code gets executed outside the browser as it is an interpreter as well as an environment for running javascript and used for server-side applications.

Browser.js is used for frontend.

Node.js is used for backend applications.

Browser.js runs any engine like Spider monkey (Firefox), JavaScript Core (Safari), V8 (Google Chrome) accordingly to the browser .

Node.js runs in a V8 engine which is mainly used by google chrome.

Bowser.js are not headless.

Node.js runs in a V8 engine which is mainly used by google chrome.

moduling is not mandatory for browser javascript.

Node.js everything is a module i.e it is mandatory to keep everything inside a module

4.What happens when you type a url in the address bar in the browser?

- You enter a URL into a web browser
- The browser looks up the IP address for the domain name via DNS
- The browser sends a HTTP *request* to the server
- The server sends back a HTTP *response*
- The browser begins rendering the HTML
- The browser sends requests for additional objects embedded in HTML (images, css, JavaScript) and repeats steps 3-5.
- Once the page is loaded, the browser sends further async requests as needed.