**Difference between Node JS and Browser (Console) JS**

Node and web browsers, both executes JavaScript, but Node does that in server side and browsers in client side. Node uses the same JavaScript engine which is the backbone of Chrome, but still we can find few differences between Node and Browser.

| **Node** | **Browser** |
| --- | --- |
| * Node doesn't have a predefined "window" object cause it doesn't have a window to draw anything. | * "window" is a predefined global object which has functions and attributes, that have to deal with window that has been drawn. |
| * "location" object is related to a particular url; that means it is for page specific. So, node doesn't require that. | * "location" is another predefined object in browsers, that has all the information about the url we have loaded. |
| * Ofcourse Node doesn't have "document" object also, cause it never have to render anything in a page. | * "document", which is also another predefined global variable in browsers, has the html which is rendered. |
| * Node has "global", which is a predefined global object. It contains several functions that are not available in browsers, cause they are needed for server side works only. | * Browsers may have an object named "global", but it will be the exact one as "window". |
| * "require" object is predefined in Node which is used to include modules in the app. | * Browsers don't have "require" predefined. You may include it in your app for asynchronous file loading. |
| * In Node everything is a module. You must keep your code inside a module. | * Moduling is not mandatory in client side JavaScript, i.e. in browsers. |
| * Node is headless. | * Browsers are not headless. |
| * Node processes request object. | * Browsers processes response objects. |

As both of them are JavaScript executor, and Node uses the JavaScript engine of a browser (Chrome), so differences are not much there. It is just the Node wrapper which has been written on top of  JavaScript V8 Runtime engine, which is deleting few objects and also including some according to the requirement of Node.

# 2 ) Summary of Video (Ryan Seddon: So how does the browser actually render a website | JSConf EU 2015)

# High level flow:

Paint

Layout

Render Tree

Parse Html

Parse css

Parsing HTML :

* HTML is forgiving my nature
* Parsing is not straight forward – it will throw error
* Can be halted
* Will do speculative parsing
* It is re-entrant.

Parsing flow:

There is tokenisation that will create parse tree and it will create dom tree and the script is executed later..

Tokenizer:

<div> </div> (open and close the tag)

<script>,<link>, and <style> tags:

It will halt the parser as a script can alter the document.

Network latency.

Link and style tags could halt JS execution.

Speculative parsing :

It will pause in the script to execute the another process with the browser and it will look ahead if we got images , scripts and css.

Reentrant :

The parsing process can be interrupted.

<script/> at the bottom :

It will parse uninterrupted

Faster to render

Defer and async attributes and Trade off

Render / Frame tree:

DOM + CSSOM:

* Combines the two object model , style resolution
* This is the actual representation of what will show on screen.
* One to one mapping of your HTML.

Mutliple Trees:

* Render objects
* renderStyles
* RenderLayers
* LineBoxes

DOM Node to render object

* Visual output
* Geometric information
* Can layout and paint
* Holds style and Computed Metrics

**Layout**

Recursive process:

* Traverse render tree
* Nodes position and size

3) Execute the below code and write your description in txt file

typeof(1) - Output : number

typeof(1.1) - number

typeof('1.1') - string

typeof(true) - boolean

typeof(null) - object

typeof(undefined) - undefined

typeof([]) - object

typeof({}) - object

typeof(NaN) - number