**1 . LIST 5 DIFFERENCE BETWEEN BROWSER JS(CONSOLE) V NODEJS**

|  |  |
| --- | --- |
| Browser JS (console) | Node JS |
| “Window” is a predefined global object which has functions and attributes that have to deal with window that has been drawn. | Node doesn’t have a predefined “window” object because it doesn’t have a window to draw anything. |
| “Location” is another predefined object in browsers that has all the information about the URL we have loaded. | “Location” object is related to a particular URL; that means it is for page specific. So, node doesn’t require that. |
| “Document”, which is also another predefined global variable in browsers, has the html which is rendered. | Node doesn’t have “document” object also, because it never have to render anything in a page. |
| Browsers may have an object named “global”, but it will be the exact one as “window”. | Node has “global”, which is a predefined global object. It contains several functions that are not available in browsers, because they are needed for server side works only. |
| Browsers don’t have “require” predefined. You may include it in your app for asynchronous file loading. | “require” object is predefined in Node which is used to include modules in the app. |

**2 . HOW DOES THE BROWSER RENDER A WEBSITE**

one will parse the HTML, one the CSS then combined to the render tree depending on which browser.

Parsing for HTML

* It is very giving by nature one can make a lot of mistakes and it will work.
* Parsing isn’t straight forward.
* In most of the languages if one make a mistake or throws an error, it will error out but HTML will try to recover.
* It can be halted, so we will go into when it can stop parsing in certain situations will do speculative parsing.
* Parsing flow there is a Tokenisation, that takes the text and turns it into what are called Tokens that will create a parse tree, that will then create a DOM tree which we all interact with java script
* Open tag will read the characters and then it says that’s the tag name looks for a closing tag and create a token called start tag, then the other end it will look for close tag

Parsing for CSS

* It will create CSS object model, like the DOM object model, basically it’s a representation of styles.

Render tree

* Combining the above 2 we got a DOM, CSS object model and that will go into and create the render or frame.
* Parsing HTML and CSS isn’t always necessary can happen in parallel Java script can effect these 2 running at the same time.
* The render tree is multiple trees Render Objects, Render styles, Render layers, line boxes.
* So the DOM node is converted to a render object, or the render object has a reference to the DOM node, it is a visual output of what is shown on the page.