1. **List 5 difference between Browser JS(console) vs Nodejs**

**Node js :**

* Node Js is known as server-side JavaScript
* Node processes request object.
* Node.js is server-side, we don't have a DOM nor do we need Cookies. Cookie's are largely implemented to track users or save user information - we're on the server side and have direct access to the database or storage resources and are using node.js to control the server itself that backs the website - this is instead of interacting with the browser that is used to communicate with our website server on the front-end.
* Able to interact and access the file system (fs)
* Node.js allows you to control the environment itself.
* Node JS supports all the new releases in Js and ECMAScript.
* In server-side JS we have the global object. This is the most publicly/globally accessible object in node.js.
* Node.js provides developers with full access to the system like any other native application.
* Node uses the CommonJS module system
* Node doesn’t have a predefined “window” object because it doesn’t have a window to draw anything.
* “location” object is related to a particular URL; that means it is page-specific. So, the node doesn’t require that.
* Of course, Node doesn’t have “document” object also, because it never has to render anything on a page.
* 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.
* “require” object is predefined in Node which is used to include modules in the app.
* In Node everything is a module. You must keep your code inside a module.

**Browser js(Console) :**

* Browser Js is known as Client-side JavaScript
* Browsers processes response objects.
* In the Browser, you are directly interacting with DOM. The DOM is a representational layer - it's a document object module that has been largely inspired by the need to create a visual user interface layer for the web via the power of browsers.
* Not able to interact with file system
* In the browser, the environment is based on the browser version.
* The browser is progressing slowly and does not support new JS concepts. We need babel support to convert the latest JS concepts to the old version
* In browser-side JS we have the window object. This is the upmost parent object in front-end/client-side JS.
* The browser sandboxes JavaScript for your safety.
* While in the browser we are starting to see the ES Modules standard being implemented.
* “window” is a predefined global object which has functions and attributes, that have to deal with window that has been drawn.
* “location” is another predefined object in browsers, that has all the information about the url we have loaded.
* “document”, which is also another predefined global variable in browsers, has the html which is rendered.
* Browsers may have an object named “global”, but it will be the exact one as “window”.
* Browsers don’t have “require” predefined. You may include it in your app for asynchronous file loading.
* Moduling is not mandatory in client-side JavaScript, i.e. in browsers.

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 that 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.