**Differences between Browser JS(console) v Nodejs**

|  |  |
| --- | --- |
| Browser JS | Node JS |
| 1) Browser has window, location, document objects to interact with DOM. | 1) NodeJS has no such objects as browser. |
| 2) Browser does not have any APIs to access file system access | 2) NodeJS provides file system access through its modules. |
| 3) Upgrade of JS features in browser are slow. | 3) In NodeJS we can use latest JS features. |
| 4) Here we have to use whatever the environment browser provides. | 4) Here we have control over environment. Means we can use whatever version of JS like ES7,6,5... . |
| 5) Here standard ES modules are implemented. Means “import” is used here. | 5) Here CommonJS modules are used. Means “require ()” is used here. |

**How does Browser render a website?**

**Flow:**

* **Parsing:** Parsing flow involves tokenizer creating parse tree which create DOM tree. Parsing is stopped when it finds <script>, <link> tags till it fetches everything from these tags. As a performance point, declaring the above tags at the bottom will not stop the parsing and increases the speed & performance while loading the page.
* ***Render Tree:***Here the combined object models (DOM+CSSOM) will be converted to Render object which is a visual output containing geometric information, styles and computed metrics which can be used to layout and paint.
* ***Layout:***In this phase browser will traverse the render object and figures out where everything should be placed. This is a recursive process, each time anything changes in the page everything is relayed out. So as a performance point, we need to do everything at once like doing all the reads at one time and writes at one time. Generally, JS frameworks will do internally to prevent re-layout.
* ***Paint:***Here the browser creates layers in incremental format. Builds up the layers which is a 12-step process like creating background color..etc.. Then creates bit map and gives to GPU which then renders actual page.

**typeof Operator**

The operator “typeof” is used to find out the datatype of a JavaScript variable.

**console.log(typeof(1)); // Returns type as "number"**

**console.log(typeof(1.1)); // Returns type as "number"**

**console.log(typeof('1.1')); // Returns type as "string"**

**console.log(typeof(true)); // Returns type as "boolean"**

**console.log(typeof(null)); // Returns type as "object"**

**console.log(typeof(undefined)); // Returns type as "undefined"**

**console.log(typeof([])); // Returns type as "object"**

**console.log(typeof({})); // Returns type as "object"**

**console.log(typeof(NaN)); // Returns type as "number"**