1. **List 5 difference between Browser JS(console) v Nodejs:** 
   1. Though Both the browser and Node.js use JavaScript as their programming language. Following are the basic differences between them

|  |  |
| --- | --- |
| Browser JS | NodeJS |
| Javascript is a programming language that is used for writing scripts on the website. | NodeJS is a Javascript runtime environment. |
| JavaScript has “window” object document Object, DOM and Web Platform APIs like Cookies on which certain operations can be done | Node doesn’t have a predefined “window” object document Object, DOM and Web Platform APIs like Cookies. |
| Javascript can only be run in the browsers. | We can run Javascript outside the browser with the help of NodeJS. |
| It is basically used on the client-side i.e. in frontend development. | It is mostly used on the server-side i.e.  in server-side development. |
| Javascript is capable enough to add HTML and play with the DOM. | Nodejs does not have capability to add HTML tags. |
| Javascript can run in any browser engine as like JS core in safari and Spider monkey in Firefox. | V8 is the Javascript engine inside of node.js that parses and runs Javascript. |

1. **summary 5 points** -<https://www.youtube.com/watch?v=SmE4OwHztCc&ab_channel=JSConf>
   1. The first step of this parsing process is to break down the HTML into tokens that represent start tags, end tags, and their contents. From that it can construct the DOM.
      * HTML Parsing - can be halted
      * Parsing will be halted as <Script>, <link> and <Style> can alter document
      * <link> and <Style> could halt JS execution
      * <Script> tag at bottom
        1. Parse uninterrupted
        2. Faster to render
        3. Defer and async attributes of script tag
      * Elements that are not in the render tree
        1. Non-visual elements like, head, script, title etc.
        2. Nodes hidden via display: none
   2. When the parser comes across an external resource like a CSS or JavaScript file, it goes off to fetch those files. The parser will continue as a CSS file is being loaded, although it will block rendering until it has been loaded and parsed
   3. Like HTML files and the DOM, when CSS files are loaded they must be parsed and converted to a tree - this time the CSSOM. It describes all the CSS selectors on the page, their hierarchy and their properties.
      * Page Layout changes immediately when user try to: -
        1. Doing font size change will re-layout the entire document
        2. Same with browser resize
        3. Accessing certain properties via javascript e.g. node.offsetHeight
   4. The render tree is a combination of the DOM and CSSOM, and represents everything that will be rendered onto the page. That does not necessarily mean all nodes in the render tree will be visually present, for example nodes with styles of opacity: 0 or visibility: hidden will be included, and may still be read by a screen reader etc., whereas those set to display: none will not be included. Additionally, tags such as <head> that do not contain any visual information will always be omitted
   5. The rendering engine traverses the render tree, starting at the top and working down, calculating the coordinates at which each node should be displayed. Painting computes bitmaps and the final step is to take that layout information and paint the pixels to the screen.
2. **Execute the below code and write your description in txt file**
   1. console.log(typeof(1));         // returns number
   2. console.log(typeof (1.1));      // returns number
   3. console.log(typeof ('1.1'));    // returns string
   4. console.log(typeof(true));    // returns boolean
   5. console.log(typeof(null));    // returns object
   6. console.log(typeof(undefined)); // returns undefined
   7. console.log(typeof ([]));       // returns object
   8. console.log(typeof ({}));       // returns object
   9. console.log(typeof(NaN));       // returns number