Question No. 1: List 5 difference between Browser JS(console) v Nodejs

Answer:

| S.No | Javascript | NodeJS |
| --- | --- | --- |
| 1. | Javascript is a programming language that is used for writing scripts on the website. | NodeJS is a Javascript runtime environment. |
| 2. | Javascript can only be run in the browsers. | We can run Javascript outside the browser with the help of NodeJS. |
| 3. | It is basically used on the client-side. | It is mostly used on the server-side. |
| 4. | Javascript is capable enough to add HTML and play with the DOM. | Nodejs does not have capability to add HTML tags. |
| 5. | Javascript can run in any browser engine as like JS core in safari and Spidermonkey in Firefox. | V8 is the Javascript engine inside of node.js that parses and runs Javascript. |
| 6. | Javascript is used in frontend development. | Nodejs is used in server-side development. |
| 7. | Some of the javascript frameworks are RamdaJS, TypedJS, etc. | Some of the Nodejs modules are Lodash, express etc. These modules are to be imported from npm. |

Question no. 2: watch & summary 5 points -

Answer:

1. From a technical standpoint, the process for loading a webpage can be broken into four stages:

navigation, parsing, rendering, and interaction. Let’s break each one of those steps down in detail.

2. Navigation: First, the browser needs to retrieve all the necessary files from a remote server to create the initial page. This is limited by factors such

as the end user’s internet speed and network latency.

It is for this reason that data centers are often spread all over the world, and CDNs are often used to deliver content from a number of potential locations,

depending on whichever is geographically closest. Files are requested over a protocol called HTTP, or HyperText Transfer Protocol.

3. Parsing:Once the browser has all the necessary files, they can begin to be interpreted. This is when the files are read and the structure of the website

begins to take shape through what’s known as lexical parsing and syntax analysis.

The Lexer breaks up code into “tokens” that can be easily processed, stripping out white space and other unnecessary characters. Each token is then passed

to the syntax analyzer to apply language-specific syntax rules and added to the parse tree. In the event that syntax errors are found, this is where a

runtime exception will be thrown.

Once lexical parsing and syntax analysis are complete, HTML and XML elements are used to create the Document Object Model (or DOM) – a series of element and

text nodes organized in a tree-like structure. JavaScript can then use these DOM nodes in order to manipulate the document’s contents.

Similar to the DOM, the CSS Object Model (or CSSOM) is also constructed at this stage, allowing JavaScript to read and modify CSS rules dynamically.

The tree-like structure of the CSSOM is what gives CSS its “cascade”, as stylesheets are interpreted from top to bottom with increasingly specific rules.

JavaScript and CSS are “render blocking” resources, meaning they can negatively affect load time by preventing the rest of the page from being parsed until

they’re finished being executed. If there’s any inline JavaScript or CSS embedded in the HTML document, it will be parsed synchronously. Since this can have

a huge impact on overall load time, loading non-critical scripts asynchronously or using the defer / async attributes is generally a good practice.

4. Rendering: Rendering is the multi-step process in which the content of the page begins to become visible to the user. This can be a relatively expensive

task for the browser to perform, depending on the complexity of the styles and animations being rendered.

First the DOM and CSSOM are combined to create the “Render Tree” by traversing the DOM nodes and finding the appropriate CSSOM rules that apply to them.

This only includes nodes that will occupy space in the layout, so if an element has display: none, it will be omitted from this tree.

Next the layout stage computes the exact size and positions of each node within the layout by creating a box model, and reserves that space on the page.

This is also commonly referred to as “reflow”.

5. Interaction: Finally, the interaction step is when the user can begin browsing and using the page. A page is considered “fully interactive” when all

previous steps have completed and users can begin to scroll, type, and interact with elements on the page.

First CPU Idle represents the point at which the page is minimally interactive – meaning it has loaded enough information for it to be able to handle a

user’s input. Most, but not all of the UI is interactive and the page responds to user input in a reasonable amount of time.

Question no. 3 :Execute the below code and write your description in txt file.

Answer:

a. typeof(1) => output: number

b. typeof(1.1) => output: number

c. typeof('1.1') => output: string

d. typeof(true) => output: boolean

e. typeof(null) => output: object

f. typeof(undefined) => output: undefined

g. typeof([]) => output: object

h. typeof({}) => output: object

i. typeof(NaN) => output: number