Q1: Difference between Browser JS(console) v Nodejs

| **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. |
| 8. | It is the upgraded version of ECMA script that uses Chrome’s V8 engine written in C++. | Nodejs is written in C, C++ and Javascript. |

Q2:

**1. Parsing**

Parsing HTML => JS and HTML is forgiving by nature. Parsing isn’t straight forward. Can be altered. Will do speculative parsing. It’s reentrant.

Parsing flow=> its cycle of Tokenizer, Tree construction, Dom Tree, script execution.

<Script>, <link>, <style> will halt the parse as a script can alter the document. =>network latency =>link & style could halt JS execution

Speculative parsing =>will look ahead. External images, scripts, css

Reentrant => means the parsing process can be interrupted

**Performance insight 1**

< /script > at the bottom => parse uninterrupted, faster to render, defer and async attributes, trade off

Css parsing =cssom

**2. Render /frame tree** => DOM+CSSOM

Combines the two object models, style resolution. This is the actual representation of what will show on screen. Not l-to-l mapping of your HTML

Multiple tree = render objects, render styles, render layers, line boxes

Not in the render tree=> non-visual elements head, script, title etc. & nodes hidden via display: none;

DOM node to render object=> visual output, geometric info, can layout and paint, holds style and computed metrics

Calculating visual properties=> combines all styles. Defaults, external, style elements & inline

Complexity around matching rules for each element. Style computation.

**3. Layout**

Recursive process=> Traverse render tree, Nodes position and size, Layout its children

Will batch layouts=> Incremental layouts. The browser will intelligently batch changes. Render tree items will flag themselves as dirty. The batch will traverse the tree and find all dirty trees. It’s Asynchronous.

Immediate layout=> doing a font-size change will relayout the entire document. Same with browser resize. Accessing certain properties via JavaScript e.g. node.offsetHeigh t

**Performance Insight 2**

Take note from the browser and batch=> Act like the browser & batch your DOM changes. Do all your reads in one pass. Followed by writes.

Real World=>Fast Dom, preventing layout thrashing. Most modern JS frameworks do this internally.

**4. Paint**

Paint setup=> will take the layout render trees. Creates layers. Incremental process. Builds up over 12 phases.

Render Layers=>creates layers from Render Objects. Position nodes, transparency, overflow, canvas, video, etc. Many-to-l relationship a Render Layer could contain multiple render objects.

Painting=> Produces a bitmap from eac layer. Bitmap is uploaded to the GPU as a texture. Composites the textures into final image to render to the screen.

**Performance Insight 3**

Inline critical CSS=> The most important bits of your site/app. Speeds up first paint times. External js & css can block. Delta last bitmap.

Q4:

1// 1 is number

console.log(typeof(1));

PS C:\Users\Lenovo\Desktop\GUVI ASSIGNMENT> node .\script.js

Number

2// 1.1 is number with decimal point

//1.1

console.log(typeof(1.1));

PS C:\Users\Lenovo\Desktop\GUVI ASSIGNMENT> node .\script.js

number

3// ‘1.1’ is string because its mentioned inside ‘ ‘

console.log(typeof('1.1'));

PS C:\Users\Lenovo\Desktop\GUVI ASSIGNMENT> node .\script.js

string

4// true is Boolean operator

console.log(typeof(true));

PS C:\Users\Lenovo\Desktop\GUVI ASSIGNMENT> node .\script.js

boolean

5// null is an object.

console.log(typeof(null));

PS C:\Users\Lenovo\Desktop\GUVI ASSIGNMENT> node .\script.js

object

6// null is Undefined.

console.log(typeof(undefined));

PS C:\Users\Lenovo\Desktop\GUVI ASSIGNMENT> node .\script.js

undefined

7// [] array is an object.

console.log(typeof ([]));

PS C:\Users\Lenovo\Desktop\GUVI ASSIGNMENT> node .\script.js

object

8// {} is an object. Regular expressions are always objects. Functions are always objects

console.log(typeof({}));

PS C:\Users\Lenovo\Desktop\GUVI ASSIGNMENT> node .\script.js

object

9// NaN is an object

console.log(typeof(NaN));

PS C:\Users\Lenovo\Desktop\GUVI ASSIGNMENT> node .\script.js

object