**1. List 5 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 Spider monkey 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.** |

**2. Watch and Summary 5 points**

**Answer:**

1. Binding
2. Rendering: Parsing, layout, painting etc.
3. Platform
4. JavaScript VM

**Parsing HTML**

1. HTML is forgiving by nature
2. Parsing isn’t straight forward
3. Can be halted
4. Will do speculative parsing
5. It’s reentrant

**Valid HTML5**

<body>

<p class=wat>My first website

<div> <span> Visitor count: 0

**Would Output**

<html>

<head> </head>

<body>

<p class=”wat”>

My first website

</p>

<div>

<span>

Visitor count: 0

</span>

</div>

</body>

</html>

**Tokenizer**

1. Tag open = **<**
2. Tag name = **div**
3. Tag close =  **>**
4. Close tag open = **</**
5. Tag name = **div**
6. Tag name = **>**

**<script>, <link> & <style>**

Will halt the parser as a script can alter the document.

* Network latency
* Link & style could halt JS execution

**Speculative parsing**

* Will look ahead
* External images, scripts, css

<script src= ‘script.js’>

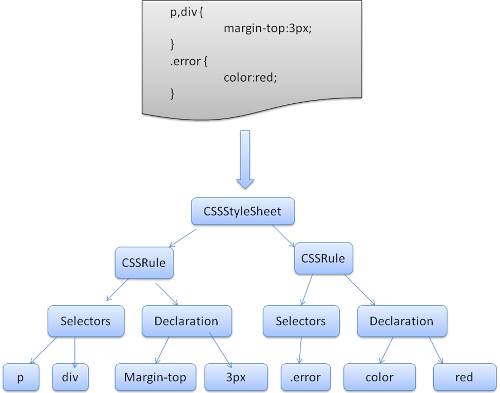
//….

<img src= ‘cat.gif’ />

<link href= ‘styles.css’ />

**<script/> at the bottom**

* Parse uninterrupted
* Faster to render
* Defer and async attributes
* Trade off



**DOM + CSSOM**

* Combines the two objects models, style resolution
* This is actual representation of what will show on screen
* Not a I-to-I mapping of your HTML

**Multiple trees**

1. Render Objects
2. Render Styles
3. Render layers
4. Line boxes

**Not in the render tree**

* Non-visual elements head, script, title etc
* Nodes hidden via display: none;

**4. Execute the below code and write your description in txt file**

* 1. typeof(1)
  2. typeof(1.1)
  3. typeof('1.1')
  4. typeof(true)
  5. typeof(null)
  6. typeof(undefined)
  7. typeof([])
  8. typeof({})
  9. typeof(NaN)

**Answer:**

let a = 1;

console.log(typeof(1));

**number**

let b = 1.1;

console.log(typeof(1.1));

**number**

let c = '1.1';

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

**string**

let d = true;

console.log(typeof(true));

**boolean**

let e = null;

console.log(typeof(null));

**object**

let f = undefined;

console.log(typeof(undefined));

**undefined**

let g = ([]);

console.log(typeof([]));

**object**

let h = ({});

console.log(typeof({}));

**object**

let i = NaN;

console.log(typeof(NaN));

**number**