ZEN class JS B27 WE day2 task:

1. List 5 difference between Browser JS(console) v Nodejs:

|  |  |  |
| --- | --- | --- |
|  | JS(console) | NodeJS |
| Execution | Output of the javascript program can be logged into the console of the browsers. | Executing js program outside a browser in a separate environment. |
| client/server side | Executing the scripts in client side using the browser console | Mostly used in server side execution of scripts |
| Integrating with HTML | Through JS in console we can manipulate and see how html tags work | We can't add or do manipulation in html tags |
| front/back end | Mostly front end development, quick checking if changes in program into the output can be observed in console | Mostly used in back end development |
| Engine | JS code written can run in any current Browser engines | V8 is the JS engine present in Node.js to run the JS part of the code. |

1. How does Browser actually render the website:

|  |  |  |
| --- | --- | --- |
| steps | Rendering process name | Process explanation |
| 1 | Parse HTML | 1. Tokenizer converts text into tokens which creates a parse tree.  2. Parse tree resultant html code after putting together all the missing tags in the code we enter  3. Parse tree output is converted into DOM tree which is what interacts with the web page. |
| 2 | Parse CSS | 1. Creation of cssom to interact with webpage  2. Cssom created by using cssrules, declarations etc. |
| 3 | Render/Frame tree | 1. Combination of dom and cssom = DOM+CSSOM  2. Actual representation of what's shown on the screen  3. Non visual elements like head, script, title are not in render tree  4. Render tree is a combination of multiple trees. |
| 4 | Layout | 1. DOM will traverse through the render tree and layouts the trees in it.  2. First will layout children  3. Instead of read and write statements mixing up, we can club all read statements first and then club all write statement next will render the webpage faster |
| 5 | Paint | 1. All information from render tree taken and data shown in the layout that have been created  2. Create layers ----> incremental painting process → 12 phases  3. Creates layers from render objects  4. Control will produce bitmap from each layer, bitmap uploaded into gpu as texture, texture converted into final image |

1. Code execution:
   1. typeof(1): number
   2. typeof(1.1): number
   3. typeof(‘1.1’): string
   4. typeof(boolean): true
   5. typeof(null): object
   6. typeof(undefined): undefined
   7. typeof([]): object
   8. typeof({}): object
   9. typeof(NaN): number