|  |  |  |  |
| --- | --- | --- | --- |
| |  | | --- | | **Node js** |  * Node doesn’t have a predefined “window” object cause it doesn’t have a window to draw anything. * “location” object is related to a particular url; that means it is for page specific. So, node doesn’t require that. * Ofcourse Node doesn’t have “document” object also, cause it never have to render anything in a page. * In Node everything is a module. You must keep your code inside a module * “require” object is predefined in Node which is used to include modules in the app. | |  | | --- | | **Browser js** |  * “window” is a predefined global object which has functions and attributes, that have to deal with window that has been drawn. * “location” is another predefined object in browsers, that has all the information about the url we have loaded. * “document”, which is also another predefined global variable in browsers, has the html which is rendered. * Moduling is not mandatory in client side JavaScript, i.e. in browsers. * Browsers don’t have “require” predefined. You may include it in your app for asynchronous file loading. |

**2.How Does The Browser Actually Render A Website**

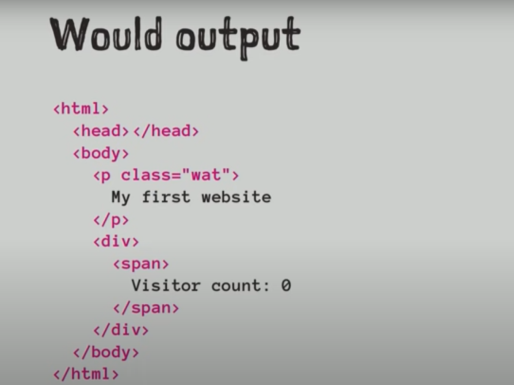
**Browser consists of**

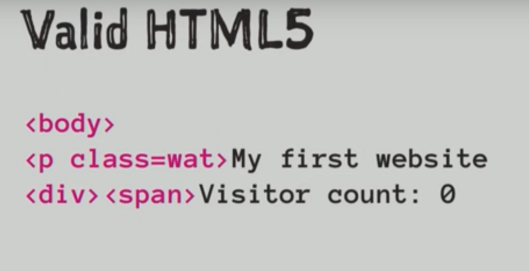
* Binding
* Rendering
  + Parsing
  + Layout
  + Painting
* Platform
* JavaScript VM

**High Level flow**

**HTML Parsing** - Creates DOM Tree

* HTML is forgiving by nature





**Parsing Flow**

* ***Tokeniser*** – Takes text and converts into Tokens
* ***Tree Constructor (Parse Tree)*** - One to one representation of HTML
* ***DOM Tree* –** interacting between Javascript in a page
* ***Script Execution* –** JS can interact with HTML and CSS and can alter the page

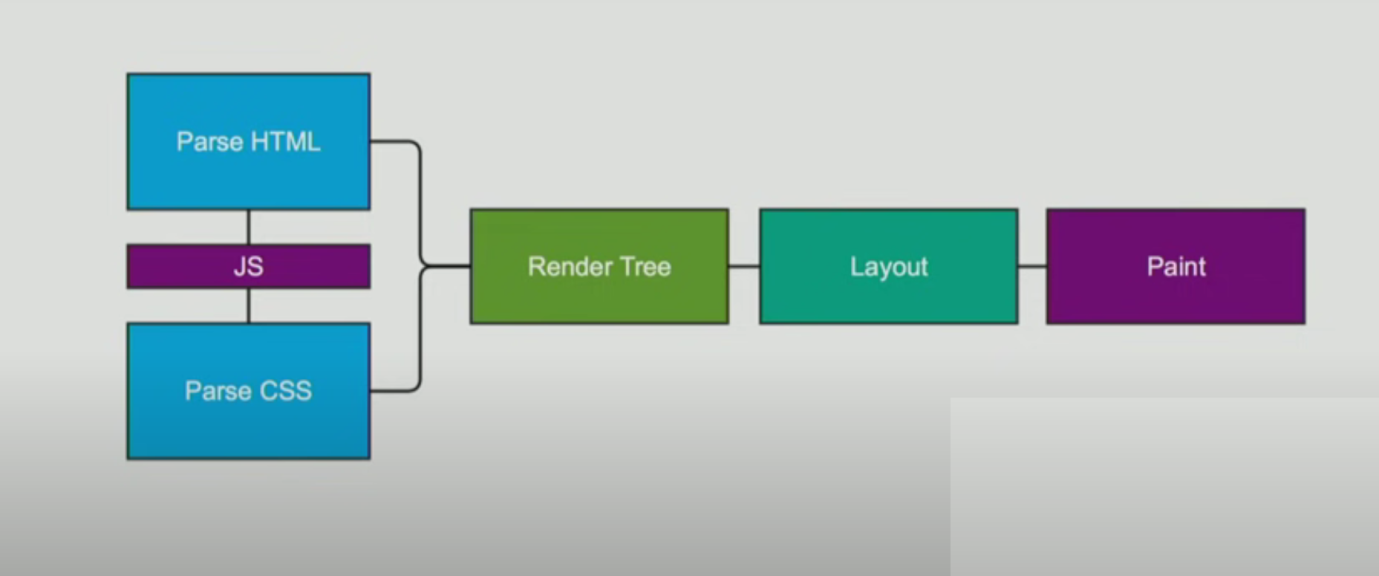
**HTML parsing** - can be halted

* <script>,<link> and <style> tags can stop HTML execution as they can alter the webpage

**Speculative parsing -** HTML parsing Will look ahead for external images, scripts, styles etc.

**Reentrant –** means HTML parsing can be interrupted by js

**CSS Paring –** Straight Forward parsing – Creates CSSOM



**Rendering or Frame Tree**

* Combination of Object Module and Style Resolution **(DOM +CSSOM)**
* This is the actual representation of what we see on the screen
* There are multiple tress in Rendering tree
  + **Render Object –** DOM Node itself
  + **Render style –** Style applied to DOM elements
  + **Render layers** – Placing elements on a page
  + **Line box** – text and its styles
* Not in render tree
  + Non visible elements – head tag, style, title etc
  + Nodes hidden by *display:none*

**DOM nodes to render Nodes :** DOM nodes are converted to render objects

* **It Consists of**
* Visual Outputs
* Geometric info
* Layout and paint
* Hold style and computed metrics

**Calculating visual properties**

* Combines all styles
* Default, external, style elements and inline
* Style computation

**Layout**

* Recursive process
* Traverse Render tree
* Layouts its child
* **Will Batch Layout** – render tree will flag itself as dirty if there is any change in DOM and the Browser will change it works as asynchronously
* **Immediate layer –** change in font size or window resize

**Paint –** will create a bitmap from render tree

**Performance Insights**

* Putting <script>,<link> and <style> tags at the bottom prevents parsing getting interrupted in middle
* Also helps in faster rendering
* Read in one go and write in one go

3.[Is it necessary to write HEAD, BODY and HTML tags?](https://stackoverflow.com/questions/5641997/is-it-necessary-to-write-head-body-and-html-tags)

Ans: Omitting the html , head , and body tags is certainly allowed by the HTML specsThe underlying reason is that browsers have always sought to be consistent with existing web pages, and the very early versions of HTML didn't define those elements.

 but generally doing so is unwise.

It has two effects - it makes the spec more complex, which in turn makes it harder for browser authors to write correct implementations

This makes the likelihood of browser errors in these parts of the spec high. As a website author you can avoid the issue by including these tags - so while the spec doesn't say you have to, doing so reduces the chance of things going wrong, which is good engineering practice.

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

typeof(1) = number

typeof(1.1)= number

typeof('1.1')= string

typeof(true)= boolean

typeof(null)= object

typeof(undefined)= undefined

typeof([])=object

typeof({})=object

typeof(NaN)= number

5.Read what is prototype

Ans:

* The prototype is an object that is associated with every functions and objects by default in JavaScript, where function's prototype property is accessible and modifiable and object's prototype property is not visible.
* Every function includes prototype object by default.
* The prototype object is special type of enumerable object to which additional properties can be attached to it which will be shared across all the instances of it's constructor function.

Example

function Student() {

this.name = 'John';

this.gender = 'M';

}

Student.prototype.age = 15;

var studObj1 = new Student();

alert(studObj1.age); // 15

var studObj2 = new Student();

alert(studObj2.age); // 15

* As mentioned before, object's prototype property is invisible. Use Object.getPrototypeOf(obj) to access prototype object
* All JavaScript objects inherit properties and methods from a prototype:
* Date objects inherit from Date.prototype
* Array objects inherit from Array.prototype
* Person objects inherit from Person.prototype

The Object.prototype is on the top of the prototype inheritance chain:

Date objects, Array objects, and Person objects inherit from Object.prototype.