1. **Difference between Browser JS (console) vs. Nodejs:**

* Node doesn’t have a predefined “window” object because 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.
* Of course Node doesn’t have “document” object also, because it never have to render anything in a page.
* Node has “global”, which is a predefined global object. It contains several functions that are not available in browsers, because they are needed for server side works only.
* “Require” object is predefined in Node which is used to include modules in the app.

In Node everything is a module. You must keep your code inside a module.

**Browser js (Console):**

* “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.
* Browsers may have an object named “global”, but it will be the exact one as “window”.
* Browsers don’t have “require” predefined. You may include it in your app for asynchronous file loading.
* Modeling is not mandatory in client side JavaScript, i.e. in browsers.

As both of them are JavaScript executor, and Node uses the JavaScript engine of a browser (Chrome), so differences are not much there. It is just the Node wrapper which has been written on top of JavaScript V8 Runtime engine, which is deleting few objects and also including some according to the requirement of Node.

1. **Watch & summary 5 points:**

* I watch about high level flow

Parse html

Js Render tee layout paint

Parse CSS

* Parsing html
* Html is forgiving by nature
* Parsing isn’t straight forward
* Can be halted
* Will do speculative parsing
* It’s reentrant
* I learn about the valid html5:
* The valid tags means the structure is not correct

Like

<Body>

<p class = watt > my first website

<Div><span>visitor count: 0

That is not prepare way there no head, html.



<Html>

<Head></head>

<Body>

<p class=”watt”>

My first website

</P>

<Div>

<Span>

Visitor count: 0

</span>

</div>

</body>

</html>

* This the prepare structure of the HTML. Each and every tags is properly open and closed

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

1. **type of(1) = number**
2. **type of(1.1) = number**
3. **type of('1.1') = string**
4. **type of(true) = Boolean**
5. **type of(null) = represents a missing object**
6. **Type of (undefined) = when a variable is declared but not initialized.**
7. **type of([]) = object**
8. **type of({}) = object**
9. **type of(Nan) = not a number**

**5. Read what is prototype**

* Prototypes are the mechanism by which JavaScript objects inherit features from one another. In this article, we explain how prototype chains work and look at how the prototype property can be used to add methods to existing constructors.
* JavaScript is often described as a prototype-based language — to provide inheritance, objects can have a prototype object, which acts as a template object that it inherits methods and properties from.
* An object's prototype object may also have a prototype object, which it inherits methods and properties from, and so on. This is often referred to as a prototype chain, and explains why different objects have properties and methods defined on other objects available to them.
* In JavaScript, a link is made between the object instance and its prototype (its \_\_proto\_\_ property, which is derived from the prototype property on the constructor), and the properties and methods are found by walking up the chain of prototypes.