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

Browser JS(console)

JavaScript is a programming language that runs in web browsers. Most websites use JavaScript and Cytobank is no exception. Most actions and state changes that happen in the Cytobank interface are governed via JavaScript.

The JavaScript console is a command line interface in your browser that can execute snippets of code. When that code snippet is designed to interact with the webpage you are currently on, result can happen that might not have been possible otherwise.

On occasion, Cytobank Support might provide you with code snippets to accomplish tasks that aren't currently possible in the Cytobank user interface.

Nodejs:

First and foremost, node.js is a javascript runtime based on chrome’s javascript engine called V8. In simple terms, people extracted the javascript engine from chrome and made it able to run standalone. But obviously it’s not that simple, there is no DOM, no UI, there are some runtime differences and we will go over the ones that affect your javascript or that are good to know.It supports all major OSes, which is a great thing because it means applications written for node.js run on all platforms. Note that you can lock yourself into a single OS in some situations when using native APIs that may not function the same or even exist in other OSesOnce installed, you will be able to start node.js with the command line, if started without arguments it will open in REPL mode where you can type javascript and it will be instantly evaluated, like in the console of a browser.The more interesting use case is when you create a javascript file and launch node.js with the file path as your first argument. It will execute the javascript code inside.

Very much like javascript running in a browser, node.js is has a single thread for running javascript and it uses the event queue. Blocking the main thread does not freeze any UI but it is still bad practice because it prevents async tasks like web requests from ever being handled. Exceptions, flow, scoping all works identical to javascript modules. But there are many significant differences.Browsers do have multiple threads to support the execution of javascript but in node.js the threadpool is used for super fast IO. When you use native modules they can tap into the threadpool, usually to do things like read from disk or send/receive data over the network in the background without wasting precious main thread CPU cycles that are reserved to your javascript code. This means that as long as you use the async versions of IO operation methods, the IO is basically zero cost for the javascript thread because the load is handled by another thread in the background. This is part of what makes it possible to have high-performance javascript based servers even though javascript itself only runs on one thread.

1. watch & summary 5 points -<https://www.youtube.com/watch?v=SmE4OwHztCc&ab_channel=JSConf>

Html is forgiving by nature. Html can be halted. Parsing Html is recntrant, Parsing html will do speculative parsing. Parsing isn’t straight forward.

Paser can be halted when script is altered the document due network latency. Link and style could halt its execution.

DOM and CSSDOM combines the 2 object models and style resolution. This is the actual representation of what will be shown on screen

1. Execute the below code and write your description in txt file
2. typeof(1)

Returns the type as Number.

1. typeof(1.1)

Returns the type as Number.

1. typeof('1.1')

Returns the type as String.

1. typeof(true)

Returns the type as Boolean.

1. typeof(null)

Returns the type as object.

1. typeof(undefined)

Returns the type as Undefined.

1. typeof([])

Returns the type as Object.

1. typeof({})

Returns the type as Object.

9.typeof(NaN)

Returns the type as Number.

4.Read what is prototype.

****A prototype**** is an existing inbuilt functionality in **[JavaScript](https://www.toolsqa.com/javascript/what-is-javascript/)**. Whenever we create a **[JavaScript function](https://www.toolsqa.com/javascript/functions-in-javascript/)**, JavaScript adds a prototype property to that function. A prototype is an object, where it can add new variables and methods to the existing object. i.e., Prototype is a base class for all the objects, and it helps us to achieve the inheritance. In this article, we will cover the following aspects of the ****JavaScript Prototype****:

A prototype is an inbuilt object where it associated with the functions by default, which can be accessible, modifiable, and create new variables and methods to it and share across all the instances of its constructor function.

****How to add variables to an object using the Prototype in JavaScript?****

As discussed, there can be a time when we need to add the new variables to an existing object, which is almost impossible in other programming languages. But in javascript, we can achieve with the help of the prototype. It follows the following syntax for adding a new variable:

****Syntax:****

ClassName.prototype.variableName = value;

### ****How to add methods to the object using the Prototype in JavaScript?****

Similar to the variables, sometimes, we need to add the methods for the existing object. The same can be achieved in JavaScript using the prototype functionality. Its syntax looks like below:

****Syntax:****

className.prototype.functionname = ()=>{

};