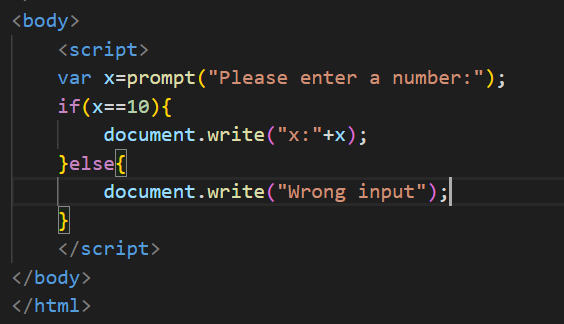
**JS** is a lightweight scripting language.

**Internal JS:**



Using <script> tag and giving the JS code inside the html file itself.

**Document:**

In the world of javascript, everything is an object. Elements, numbers, arrays, strings, document all are considered as objects.

document.write(“sample”); Here document is an object and write is the method.

This will print whatever you give in the method in the web browser.

**Commenting JS code:**

Same like in JAVA,

// - comment a single line

/\*

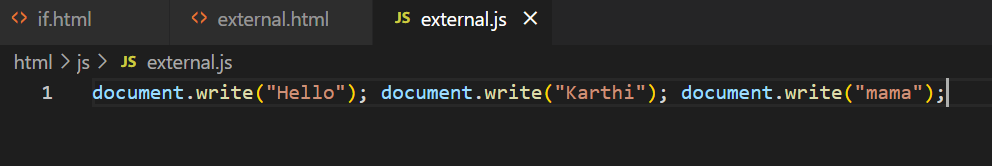
\*/ - comments multiple line

**HTML code:**

<!-- --> - comments both single and multiple line of HTML code.

**Semicolon:**

We don’t have to use semicolon for every statements in JS. Only when we use multiple JS codes in a single line then we need to use semicolon, else it will cause error.



But it is always a good practice to use semicolon for all statements in JS.

**Variables:**

JS has three primitive datatypes, which are numbers, string and Boolean.

Defining a variable: var x=5; var is the keyword used to define variable. Unlike other programming languages, we don’t need to mention datatype for variable. JS will automatically checks the data and identify its DT at runtime.

That’s why JS is called untyped or dynamic language.

Naming conventions in JS:

Do’s: Don’ts:

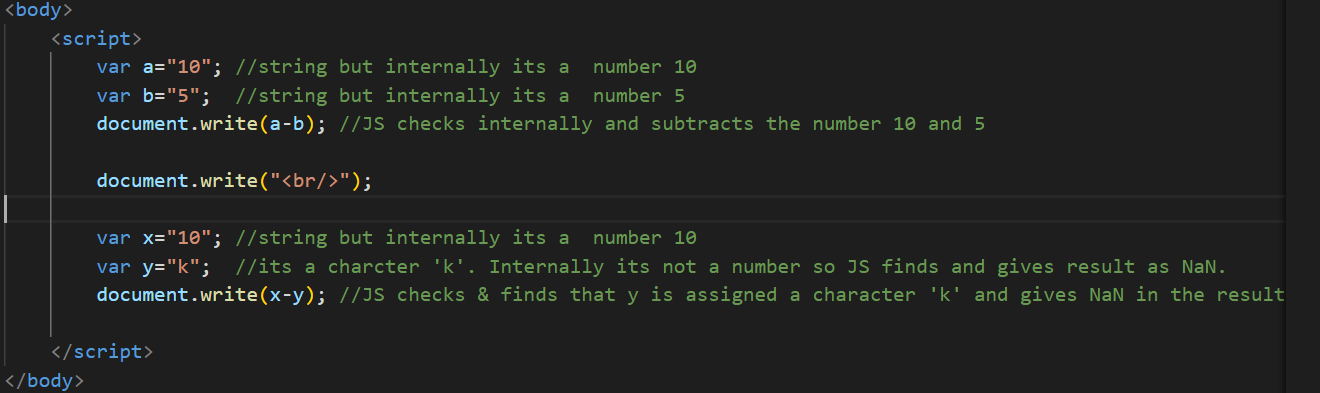
\_variable :– can start with underscore. 10variable – shouldn’t start with number

variable :- can start with alphabet

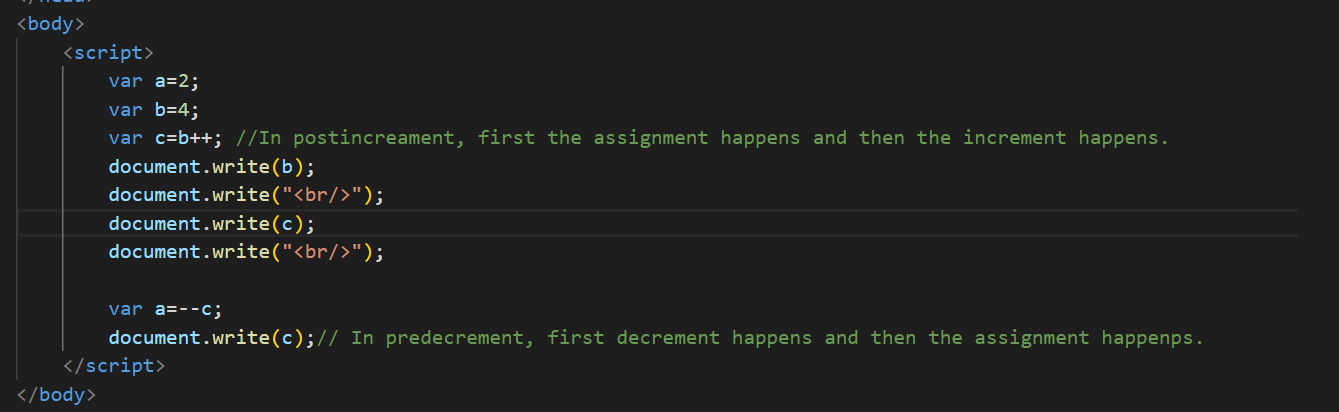
Note: variables in java are case sensitive. And variable name should not be reserved words like break,if and for etc..

**NaN(Not a Number):**

NaN is a inbuilt value in JS. When we subtract a number with a character, JS finds out that one we are subtracting with a character and it gives NaN in result.



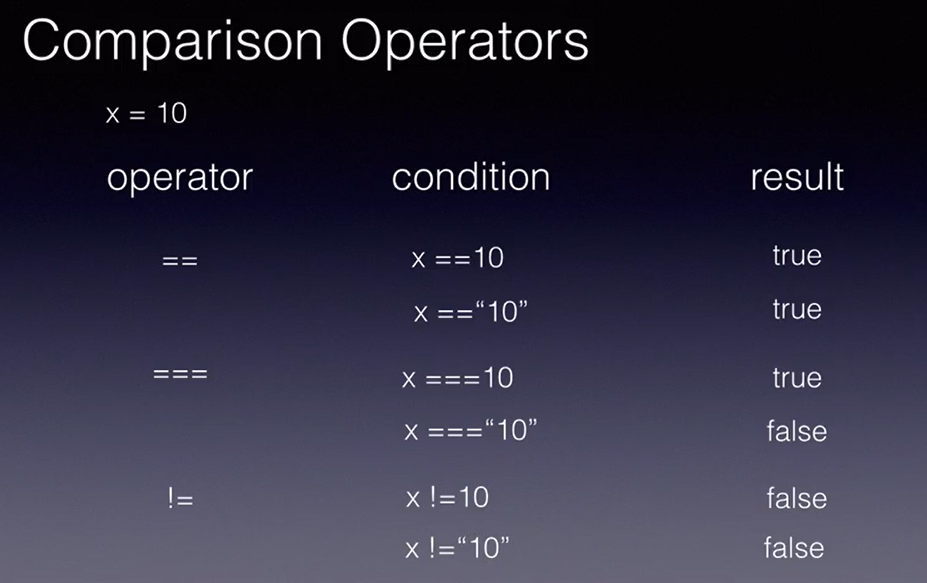
**Increment & Decrement:**

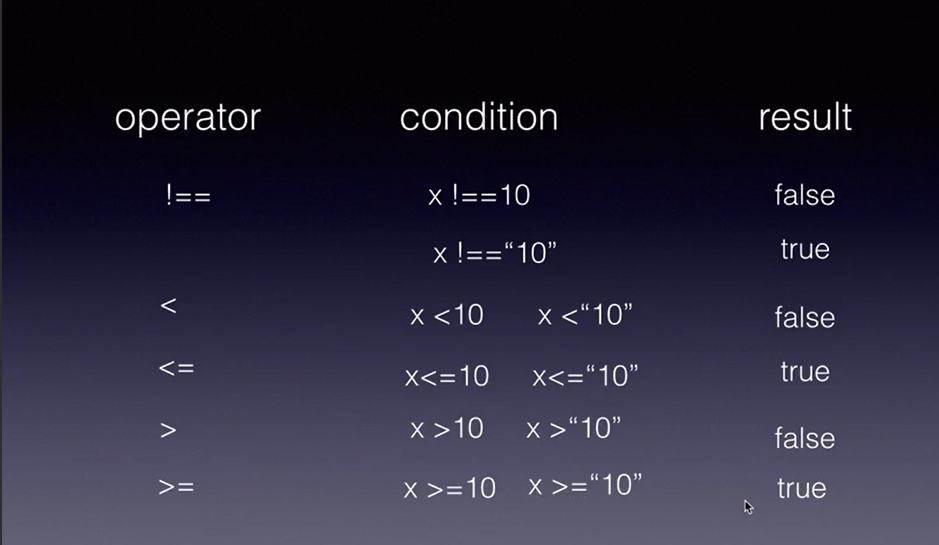


In post increment/decrement, first the assignment happens and then the increment/ decrement happens.

In pre increment/decrement, first increment/decrement happens and then the assignment happens.

**Comparison Operators:**

****

****

In JS, unlike other languages, the usual comparison operator(==) does not check the datatype of variables. If x is 10 and we are comparing it with y which is “10”(String), it will return true as both are looking same.

So if we have to compare by checking the datatype, then we have to use ‘===’ operator.

Likewise, != just checks the variables by appearance and !== checks the variable with their datatype.

**Pop-up:**

3 types of button:

1.alert – OK button

2.confirm – OK and Cancel button

3.prompt – gets input from client and can perform functions with that data.

**Conditional Statements:**

Conditional Statements(if else and switch) are same like in java. Refer VS Code for JS conditional statements.

**Loops:**

Loops(for,while and do while) are same like in java. Refer VS Code for JS conditional statements.

**Functions:**

Function is also almost same as in Java. Except we don’t mention the datatype for parameters and we don’t mention the return type(datatype). We just mention function functionName.

Syntax of a function:

A close-up of a computer screen

Description automatically generated

Function with arguments:

A close-up of a computer screen

Description automatically generated

**Closure:**

A function within a function is called closure.



**Events:**

Note: In JS, tag is called as object. Also as element.

**onmouseover and onmouseout:**

Hovering, clicking are the events. And we can handle them like what needs to happen when it is hovered or clicked.

A screen shot of a computer code

Description automatically generated

In the above example, ‘this’ keyword is a inbuilt keyword that takes the element/object/tag(<p>) and passes it to the function. So that we can alter the contents of this tag in the function.

Refer VS Code Events folder for all the events topic(onclick(),onchange(),onfocus,onblur,onkey,onload,setTimeOut,onSubmit).

**Built-in objects:**

Refer VS Code Inbuilt Objects folder(explanation in comments).

**Error Handling:**

In JS, we use the term error instead of exception.

There are two types of errors in JS,

1. Syntax error 2. Runtime error

Handling error using try catch block:

It is almost similar to Java. Just that Instead of Exception class we need to catch the error in a obj and print it in catch block.

A computer screen shot of text

Description automatically generated

Note: obj keyword in catch block can be anything. You can even give just ‘ob’. The keyword we give in catch block will automatically catch the error.

Finally keyword:

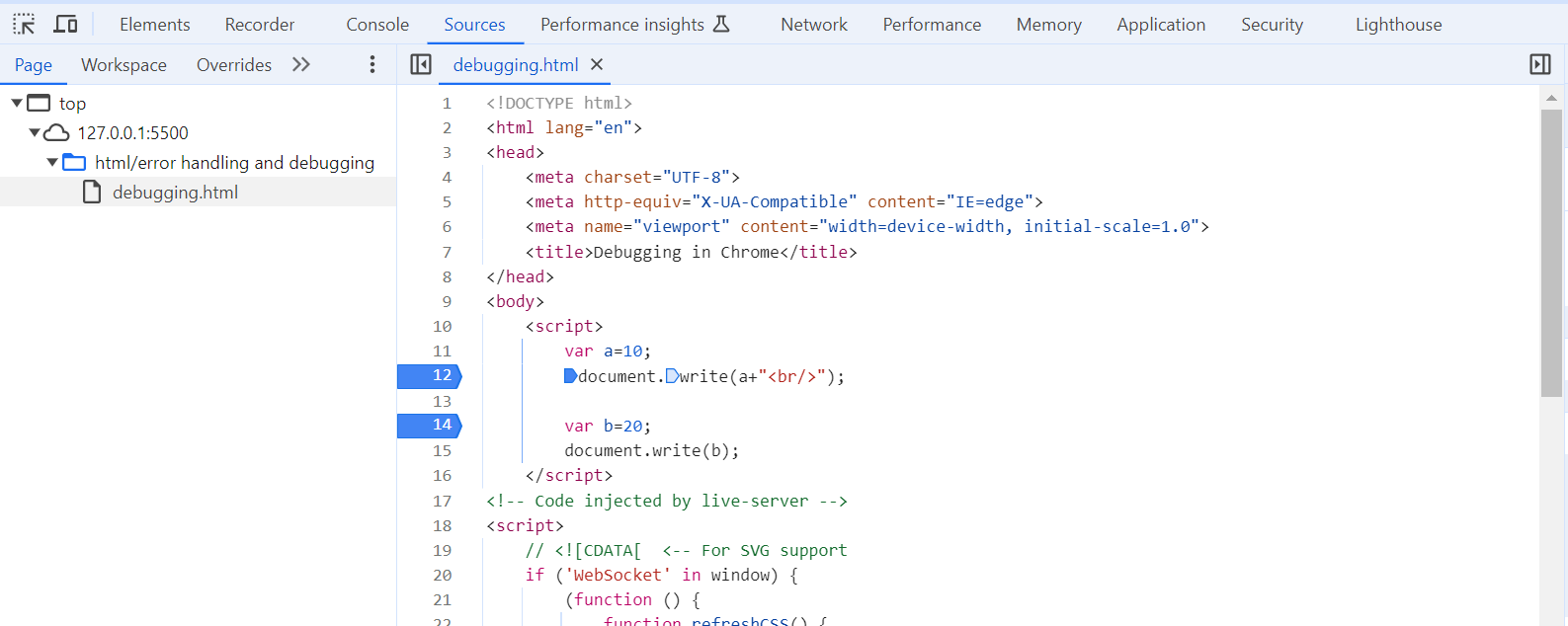
A try block shld have either a catch or finally block or both catch & finally block.

finally block will get executed even if there is error in try block or not. But catch block will get executed only if there is error.

Debugging in JS:

If you want to debug, then right click -> inspect -> Sources -> Click on the html file name in the left side.

Setting breakpoints: Clicking the numbers will make it a breakpoint.

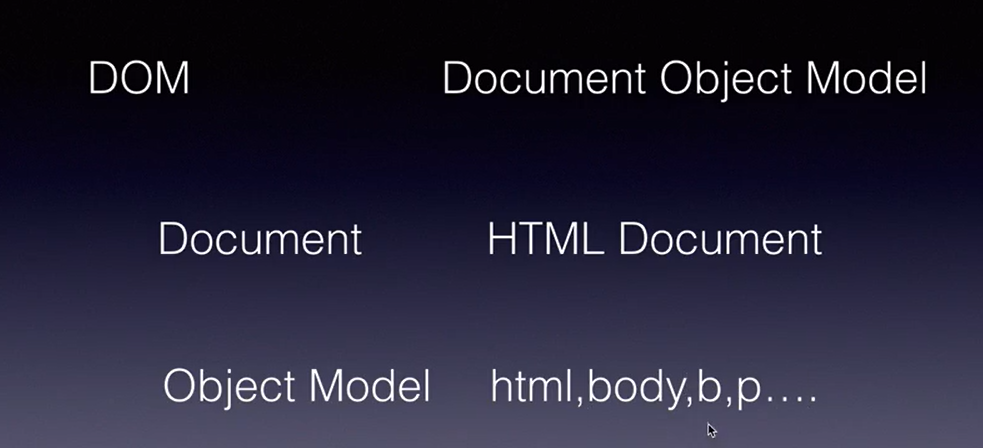


So when you reload the html in browser, it will get executed till the breakpoint and stop there. You can give resume or step over next function in the options available on top right side.



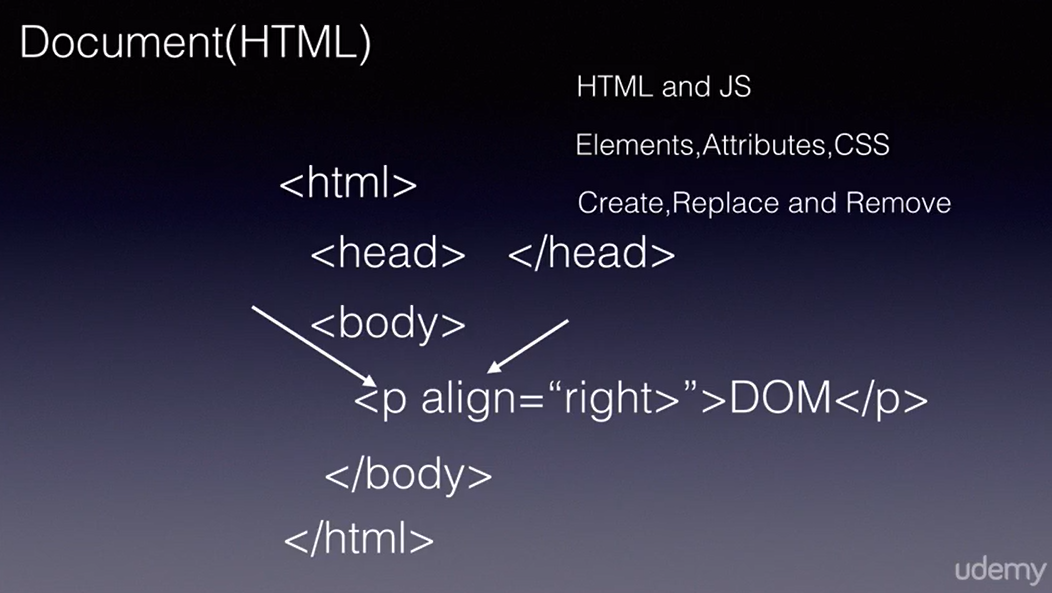
**DOM(Document Object Model):**

DOM is a API used in JS to access & manipulate the tags of HTML.



Document – HTML Document(When the HTML doc is loaded, it gets saved in memory).

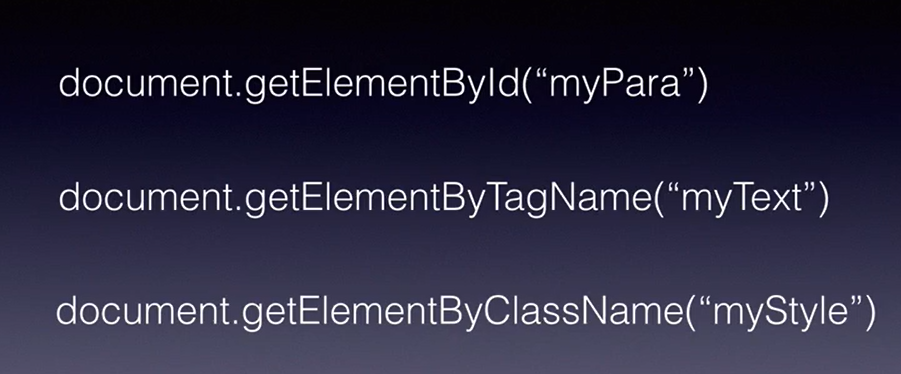
Object Model – The entire HTML doc is represented in the form of objects(i.e) all the tags in HTML like html,body,p,b ect is represented in the form of objects.



JS uses DOM API to access all the tags/objects in HTML.

Using DOM, JS can access all the elements, attributes and also the CSS(styling) of the HTML and create, replace and remove those tags/objects in HTML from JS itself.

There are several inbuilt methods to achieve this,



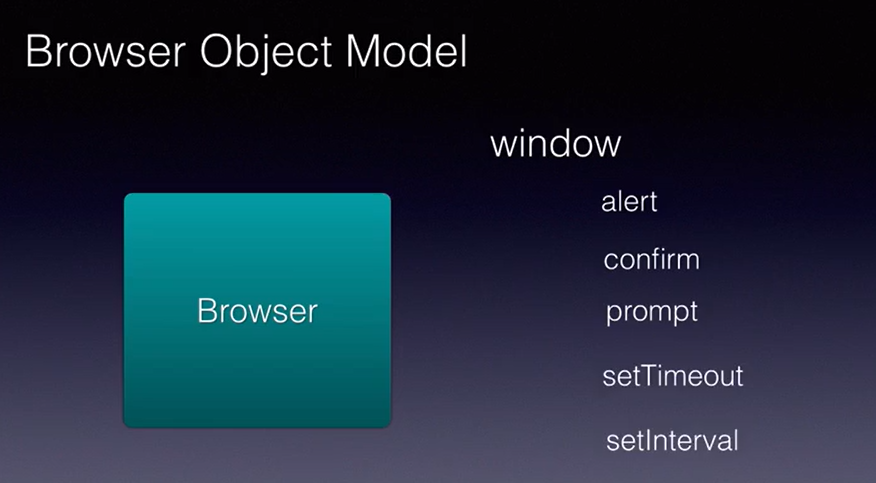
**Nodes:**

In JS DOM API, HTML doc is represented in the form of nodes. Parent and child nodes. We can create remove replace the child nodes with the help of DOM API.

**HTML BOM(Browser Object Model):**

BOM allows us to use diff functionalities of the browser in the form of objects.

In JS, browser is represented in the form of ‘window’ object.



These are the functionalities we did using window object. These alert, confirm …setInterval all are called child elements/objects. These child objects name may differ in some browsers but mostly it will be the same.

We will know see these 3 below functionalities of window obj.

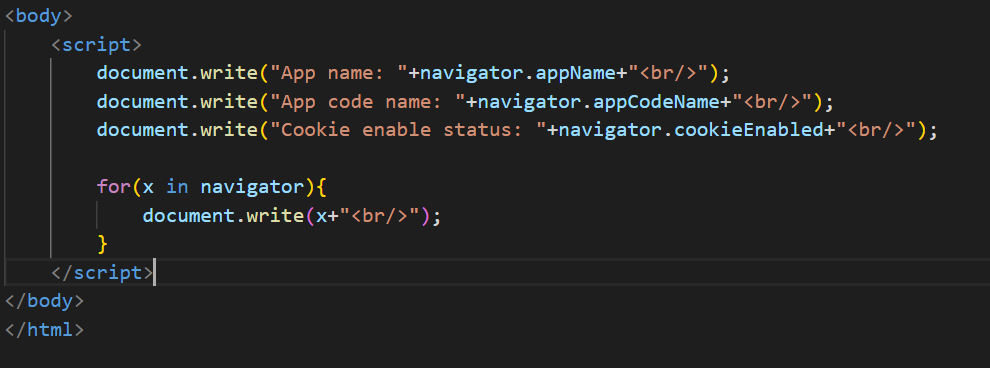
A screenshot of a computer

Description automatically generated

**Navigator:**

To get browser info. For example, we can check if cookie is enabled or not in the browser using navigator.

Also, for in loop is used here. It is like for each loop of java.



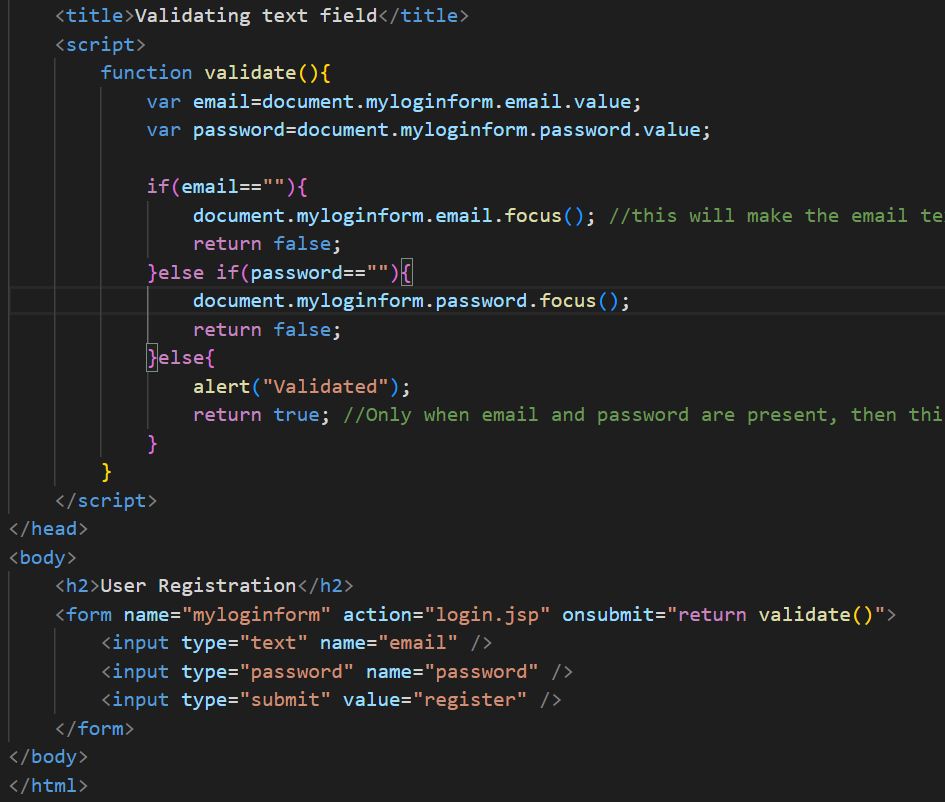
**Note:**

Name attribute just gives the name for that element.

Value attribute will give the value inside that element as value. Eg: If u give value attribute in a text input element and give a name in value attribute, that name will be displayed inside the text field in browser.

**Form Validation:**

**Text Field:**

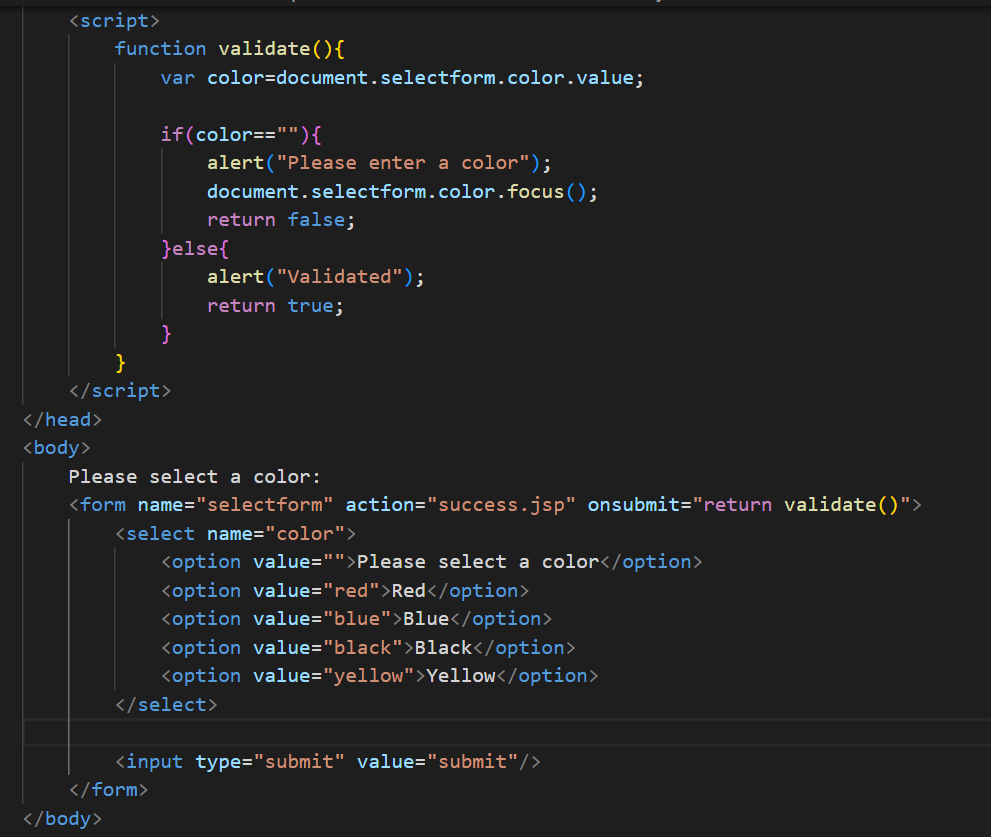
****

**Validating Numbers got from user:**

**A screen shot of a computer program

Description automatically generated**

**Validating a dropdown/select:**

****

**Regular Expression(RegEx):**

/ / - RegEx pattern shld be inside this slash

[ ] - Regex pattern will be inside this square brackets.

^ - Represents that the start of the input shld be according to the condition given inside sq brackets.

$ - Represents that the end of the input shld be according to the condition given inside sq brackets.

‘+’ – make sure that the input shld have atleast one to any no of character or digit according to the condition; min one max any no of letters/digits.

‘\*’ - make sure that the input shld have zero to any no of character or digit according to the condition; min zero max any no of letters/digits.

For more information, refer W3 schools.

**Validating text:**

**A screen shot of a computer program

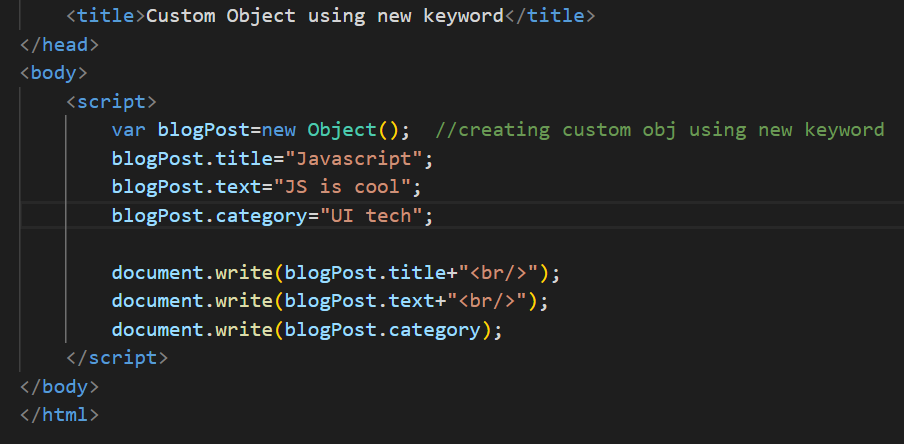
Description automatically generated**

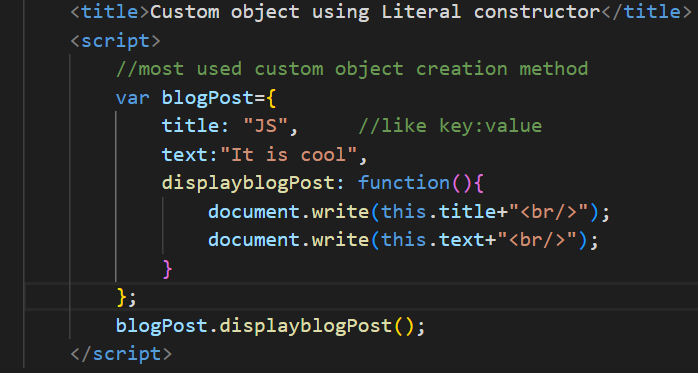
**Validating alphanumeric:**

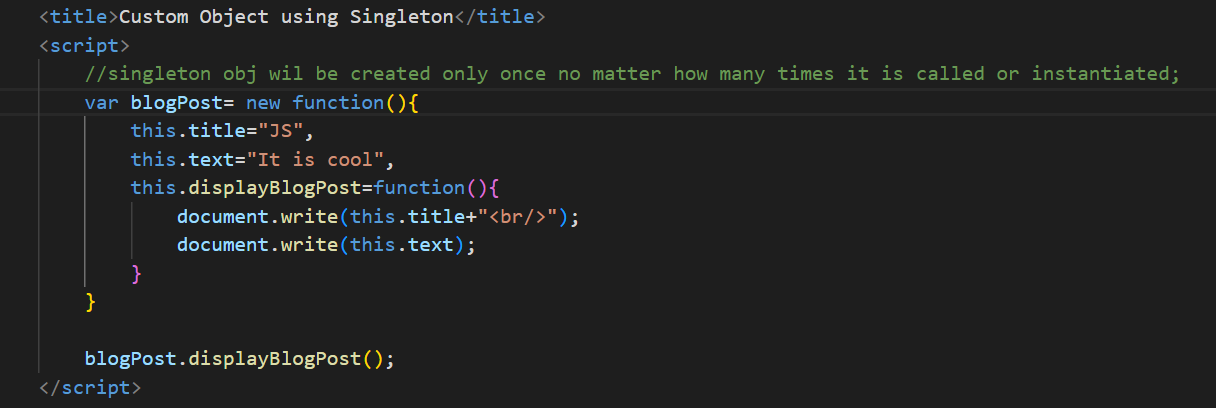
**A computer screen shot of a program code

Description automatically generated**

**Objects:**

****

****

****