# JAVASCRIPT

## NodeJs

* Used in VS Code terminal with the keyword **node** and javascript name for running the .js file
* Type **node –version** in command prompt for ensuring nodejs is installed in the PC

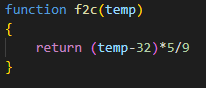
## Datatypes in Javascript

* **string:** Anything in between “ ”or ‘ ‘
* **numbers:** Any numbers
* **boolean:** Values that are either “true” or “false”
* **object:** Anything that comes between {key: value}, similar to dictionary in python
* **array:** Any values between [value1, value2], similar to list in python
* **undefined:** If a variable is declared without a value, it is mentioned as undefined
* **null:** It is a void value used to keep a variable from being undefined
* **symbols**
* **bigints**

## Operators

* **Arithematic** : + - \* / %
* **Relational** : == >= <= > < !=
* **Logical** : && || !
* **Boolean** : & | !
  + Example: 5 & 4 = ?
  + 5 in binary is 101
  + 4 in binary is 100
  + 101 & 100 is 100, it take logical AND on corresponding unit places
  + So 5 & 4 = 4
* **Assignment** : += -= \*= /= %=
* **Shift** : >> <<
  + 5>> = ?
  + 5 in binary is 101
  + So the right shift operator shifts the binary values towards the right and adds a 0 on the left
  + So 101 becomes **0**10
  + 13<<2=?
  + 13 in binary is 1101
  + Left shift operator shifts the binary operator to the left. But here its says <<2
  + So it adds two 0’s one the right making it 1101**00** which is equal to 52

## Commands in JavaScript

* **console.log(“ ”)** :
  + Used for displaying items in console section of website
  + Similar to **print(a,b)** statement in python
* **var**:
  + Variable declaration operator that affects a variable globally
* **let**:
  + Variable declaration operator that affects a variable locally
* **const**:
  + Variable declaration operator used to keep the value constant throughout the script
* **function**:
  + It is a keyword used to initiate a function inside JavaScript
  + 
* **class**:
  + It is a blueprint of an object
  + The objects in the class is created by the keyword **constructor**
  + The properties or arguments of a **constructor** is called by the syntax **this**.var\_name=argument\_name;
  + An object is declared within the body section of the script is using the keyword **const** object\_name = **new** class\_name(argument\_values);

## Conditional Statement

* **if () {…}**
* **if () {…} … else {…}**
* **if () {…} … elseif {…} … else{…}**

## Loops

* **for-loop**
  + **for-of-loop:** Used for arrays [ **for**(*let* var\_name **of** array\_name {…}) ]
  + **for-in-loop:** Used for objects [ **for**(*let* var\_name **in** object\_name {…}) ]
  + **for-each-loop:** Used to iterate over arrays [ **for**(datatype var\_name **:** array\_name {…}) ]
* **while-loop**: [ **while**(condition) {….} ]
* **do-while-loop**: [ **do**{….}**while**(condition) ]

# DOM

## Document Object Modelling

* DOM is a modelling concepts where all the HTML tags used are considered as **nodes** and these create a root-branch type relation similar to that of a tree
* DOM is effective in manipulating the contents of a HTML documents easily
* **class** & **id** is used to acknowledge the **nodes**
* Common ways to select the nodes are by using *selectors*
  + *id*: **getElementByid(“**id\_value**”)**
  + *class*: **getElementsByClassName(“**class\_value**”)**
  + *tag*: **getElementsByTagName(“**tag\_name**”)**
  + *custom CSS*: **querySelector(“**#id-value *or* .class-name**”)** or **querySelectorAll(“**#id-value *or* .class-name**”) NB:** For **querySelectorAll(),** we have to select elements as array (E.g z[1].innerHTML=”<h1>Hello</h1>”)

## Scripting

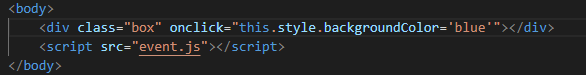
* **<script></script>** tag is used for linking HTML document to JavaScript
  + **src** attribute is used for mentioning the name of the .js file
* Inside JavaScript document, we have to declare a variable for each **node**

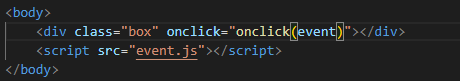
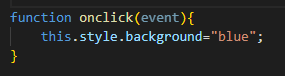
****

* Now that the **node** has been addressed, we can manipulate the **node elements** using node\_variable\_name . node\_element = “ ~change~ ”; 
* Remember the **<script></script>** tag can only be used in the end portion of the **<body>** tag so that the changes can be accessed once after the body contents are compiled.
* variable\_name.**innerHTML:** Helps to access and manipulate and entire tag and its contents
* variable\_name.**innerText:** Helps to access and manipulate just the text contents alone within a defined id or class or tag.
* variable\_name.**style.color=””**: To set the color of the text
* *const* variable\_name=document.**createElement(“***tag***”) :** To create a new tag using DOM. The contents can be added by the code -> variable\_name.innerTEXT= “Text”;
* document.body.**appendChild(**variable\_name**);** It is used to add the elements created by **createElement()** to the body of the document. If you want to append the new element inside a specific element, you can do that by document.*getElementById(“*id\_name*”).***appendChild(**variable\_name**);**
* document.**write(“ ”)**: To print a statement inside the double quotes
* *let* input\_variable = variable\_name.**getAttribute(“***class\_name***” or “***value***”):** Gets the attribute value of any attribute of any specific tag. For E.g: <a class=”homepage”> and if you type var x=document.**getByElementTagName(“a”)**; let y = x.**getAttribute(“**class"); y returns the value “homepage”
* variable\_name.**removeAttribute(**“Attribute”**) :** Removes the attribute from the particular tag

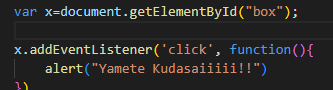
# EVENTS

## Types of Events

* **Inline-Events:** ****
* **Internal-Events:**

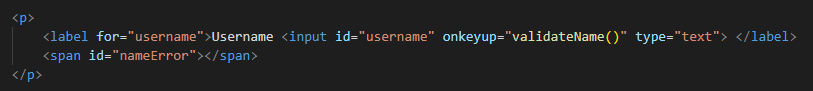
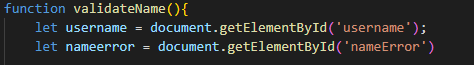
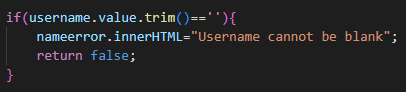
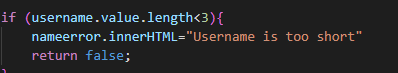
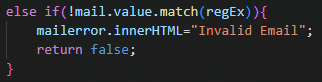
**** 

* **External-Events:**

* In this case, similar to DOM we have to declare an element\_name and use element\_name.**addEventListener(“*event*”,function(){…})** for creating an event. This method is commonly followed because it keeps the HTML document much more neat
* **event.target**.***property*** is same as **this.*property***
* If we set function(*event*){…} and insert console.log(*event*) inside the function, it can be seen while inspecting the webpage, inside console->event.

## VALIDATION

* For validation of a form, it is necessary for an input tag to have an **id** & **event** (inline, internal or external)**.**
* Usually a *function()* is passed in the **event** to execute when the event occurs.
* To display the return message, a **<span>** tag with an unique **id** should be provided. 
* Then a JavaScript is used to mention all the functions and statements just like all other coding languages. But since this is a scripting language, it only works with a front-end language
* ***function* function\_name (){….}** is written inside the JavaScript with declaring the element\_variable\_name either inside or outside the function. 
* After declaring a variable\_name for both **<input>** & **<span>** tags, We can set conditions using **if(){…}, else if(){…}** or **else(){…}** statements.
* To obtain a condition out of a variable\_name we use keywords such as
  + variable\_name**.value** : Returns the value of the element tag (Not attribute)
  + variable\_name**.value.trim()** : Returns the value of the element tag without blank spaces
  + variable\_name**.value.length** : Returns the length of the string value. 
  + variable\_name.**value.match(***variable***) :** Returns *true* if the content in the variable\_name matches *variable*. 
* Then to return the display message corresponding to the condition, set span\_variable**.innerText**-or-**innerHTML=”***text***”;**
* You can also return a Boolean variable to activate certain conditions together, just type *return true/false;* along with above statements.
* You can also use *return function\_name()* in **eventhandlers** to activate it only when certain functions return *true* value. 
* For emails, to set a certain format such as \*\*\*\*\*\*@\*\*\*\*.com , you can search **regexr.com** in internet to set various email validation conditions. 