Bro Code - JS and Rahul Shetty - JS

29 October 2024

12:51

28 October 2024

12:43

* JS file, HTML file and CSS file.
* Configure CSS file in HTML using <link *rel*="stylesheet" *href*="style.css"> in <head>.
* Configure JS file in HTML using <script *src*="index.js"></script> in <body>.
* Install extension - Live Server in VSCode and right click HTML file -> open live server.
* Providing attributes in HTML as <h1 *id*="title"></h1> and configuring the values from JS file

document.getElementById("title").textContent = `The execution is getting started`;

* \*\* in JS stands for exponential operator.
* Math class has a number of methods useful for working with numbers.
* A two dimensional array is multiple arrays inside a common array as show below

Machine generated alternative text:
const matrix — [[1, 2, 3],
[4, 5, 6],
[7, 8, 9]];

* JSON.Stringify() - used to convert a JSON object to a JSON string.

Machine generated alternative text:
const names . [“Spongebob’, “Patrick”, “Squidward”, “Sandy”];
const jsonString — JSON.stringify(names);
conso1e.1ogsonStrin);

* JSON.parse() - used to convert a JSON string to a JSON Object.
* Fetch method can be used to navigate the JSON response. In the below example, response.json() returns a promise object hence it is followed by another then().

Machine generated alternative text:
fetch(”people . json’)
.then(response —> response.jsoni)
.then(values —> values.forEachÇyalue —> conso1e.1og(va1ue)])

* In a callback function, it is not mandatory that the parameter name must be 'callback'.
* While working with functions like forEach, map, filter, reduce which accepts a callback. To improve understanding it is better to create a separate function and call it within as below. Thus the syntax would remain simple.

Machine generated alternative text:
const ages = [16, 17, 18, 18, 19, 20, 60];
const adults = ages.filter(isAdult);
console. log(adults);
function isAdult(element){
j return element >= 18; __________

* 'THIS' keyword acts as the bridge or connectivity between the object and the values being passed. So that the values can be accessed outside the constructor.
* Static keyword is assigned with Class and not objects. The optimal candidates to be assigned static are the ones whose values doesn't change with objects.
* Super keyword can be used to call the parent class constructor. In the below example the common values used by multiple classes that inherit the parent class - Animal such as name and age are moved to the parent class to avoid repetition. While calling 'super' do pass the arguments.

Machine generated alternative text:
class Animal{
constructor(name, age){
this.name • name;
this.age . age;
)
)
class Rabbit extends Animal{
constructor(name, age, runSpeed){
super(me, age);
this.runSpeed • runSpeed;
}

* In the below example we are using the super keyword to invoke the parent class method - move.

Machine generated alternative text:
move(speed){
console.logÇîhe ${this.name} moves at a speed of ${speed)mph’);
)
)
class Rabbit extends Animal{
constructor(name, age, runSpeed){
super(name, age);
this.runSpeed — runSpeed;
H
runQ{
console.log(’This ${this.name) can run’);
super. movejthis. runspee4);
)

* Getters and Setters are not used just to implement encapsulation. The purpose of getters and setters is it can be used to create additional conditional checks before setting and getting values.
* Accessing nested object values.

Machine generated alternative text:
const person • (
fullNa.e: Spongebob Squar.pants,
age: 3e,
isStuderit: tru,
hobbies: (karate, jellyfishing, cooking9,
address: (
stret: 124 Conch St.,
city: eikini Botto.,
country: Int. Water
)
for(const property in perso.address)(
5 console. log(person.address[property);

Machine generated alternative text:
1 .
2 class Person(
3
4 constructor(n, age, .. .addr.ss){ _______________________________
5 this.na.e • na;
6 this.age • age;
7 this.address • new Address(. . .address);
8 )
9)
11 class Address(
12 ____________________________________
13 constructor(street, city, country)(
14 this.street • street;
15 this.city — city; P
16 this.country • country;
17 )
18 )
19
20 const perSonl — new PersonÇspongebob, 30, 124 Conch St.,
21 ßikini Botto.1,
22  Iatrs);

* While handling an array of objects. We can use the array functions - push, pop, slice, map, forEach, reduce, filter etc to work on these objects.
* Sort function, sorts the array lexicographically. Hence numbers wont get sorted rightaway for that we have to use the following -

ascending

Machine generated alternative text:
let numbers = [1, 10, 2, 9, 3, 8, 4, 7, 5, 6];
numbers.sort(Ja, b => a - b);

descending

Machine generated alternative text:
let numbers = [1, 10, 2, 9, 3, 8, 4, 7, 5, 6];
numbers.sort{(a, b) => b - aI;
‘‘‘

* We can also sort elements in object based on a specific property value

Machine generated alternative text:
JS ¡ndex.js >
1 II sort() = method used to sort elements of an array in
2 II Sorts elements as strings in lexicographic ‘
3 II lexicographic • (alphabet + numbers + symbo
5 const people — [(name: “Spongebob”, age: 30, gpa: 3.0),
6 (name: “Patrick”, age: 37, gpa: 1.5),
7 (name: “Squidward”, age: 51, gpa: 2.5),
8 (name: “Sandy”, age: 27, gpa: 4.0)]
le people.sort((a, b) —> a.age - b.ag)
12 console.log(people);

* Whereas to sort the String property value in an object we need to use localeCompare method

Machine generated alternative text:
JS ìndex.js X (> ndex.html # style.css .
JS indexjs > ...
1 // sort() = method used to sort elements of an array in place.
2 /1 Sorts elements as strings in lexicographic order, n
3 II lexicographic = (alphabet + numbers + symbols) as s
4
5 const people — [(name: “Spongebob”, age: 3e, gpa: 3.0),
6 (name: “Patrick”, age: 37, gpa: 1.5), 4uJ
7 J (name: “Squidward”, age: 51, gpa: 2.5),
8 (name: “Sandy”, age: 27, gpa: 4.0)]
9
le people.sort((a, b) —> a.name.localeCompare(b.name));
12 console.log(people);

* Shuffle array elements using the Fisher-Yates algorithm.

Machine generated alternative text:
JS index.js X <> index.html # style.css
JS index.js >  shuffle
1 II Fisher-Yates algorithm
2
3 const cards — [‘A’, 2, 3, 4, 5, 6, 7, 8, 9, 10, OI, °Ir’, °K’];
4
5 shuffle(cards);
6 _______________ ____________ _____________
7 console.log(cards);
8
9 function shuffle(array)(j
le for(let i — array.length - 1; i > 0; i--){
11 const random — Math.floor(Math.random() (i +
13 [array[i], array[random)] - [array[random], array[i]J;
15 

* Closure in Javascript is the process of creating one function inside another function. But in detail closure is actually an object that references multiple functions within one function so as to protect the variables from being public in nature.
* toLocaleString() method can be used to format the currency as per standards of different countries.

Machine generated alternative text:
13 1umber.toLocaleString(”en-US”, (style: “currency”, currency: “USD”});
14 nbertoLocaleString(”hi-IN”, (style: “currency”, currency: “INR”));I
16 number); .
_n _

* ES6 module - an external file that contains reusable code that can be imported into JS files. The module will contain variables, functions, classes etc. Whatever method or variables or classes needs to be exported we need to prefix those with export keyword. At first we need to specify the JS file to which we are importing to have type ="module".

Machine generated alternative text:
<body>
<script type.”module’ src.”index.js”></script>I
<I body>
</ html, .

Machine generated alternative text:
iS index.js iS mathutitjs X N stybsxss
iS rn.thUbLs >  getvolume
1
2 export const PI — 3.141S9;
3
4 export function getcircu.f.r.nc.(r.dius){
S return 2 • PI • radius;
6)
7
8 export function g.tArea(radius)(
9 return PI • radius • radius;
le )
Il
12 export IFunction getVolt.(radius)(
13 return 4 • PI • radius • radius;
14 )

* We can use the import keyword along with object destructuring to import the same into another JS file.

Machine generated alternative text:
35 nciex.js x # seyes ‘
JSindeicjs>...
1 iaport (PI, getCircifcrence, getArea, g.tvolu.e) fro. ./.athutil4s;
2
3 consol..log(PI); __________________
4 const circueferenc. • getCircu.f.r.nc.(10); .
S const area • g.tAi.a(1O);
7 consol..log( S(circu.f.r.nc..toFix.d(2))c.);
8 console. log( ${ar.a.toFix.d(2))c.’2);
__________________________________________ F

* DOM elements can be accessed using the below functions

Machine generated alternative text:
JS index.js
1 II
2 II
3 II
5 II 1. document.getElementByld()
6 II 2. document.getElementsClassName()
7 II 3. document.getElementsByTagName()
8 II 4. document.querySelector()
9 /1 5. document.queryselectorAll()
fi ELEMENT OR NULL
// HTML COLLECTION
// HTML COLLECTION
Ii ELEMENT OR NULL
fi NODELIST
element selectors Methods used to target and manipulate HTML elements
They allow you to select one or multiple HTML elements
from the DOM (Document Object Model)
E’

* DOM elements can be navigated using couple of functions

Machine generated alternative text:
JS index.js > [e] element
1 /1 DOM Navigation — The process of navigating throu h the
2 /1 of an HTML document using
3
4 /1 .firstElementChild
S II .lastElementChild
6 /1 .nextElementSibling
7 // .previousElementSibling ‘.
8 II .parentElement
9 /1 .children
q ii

* A

**Javascript beginners Rahul shetty**

* Download visual studio code
* Create a file basics.js
* Console.log("hello world") to print and save it.
* Give cmd in vs code terminal - node basics.js to run the js file
* // for typing any comments on single line and /\*… /\* to write multiple line comments.
* In JS variables can hold any value unlike in java. For example we can give as var a =4.
* Until es5 JS engine we could use only var but from es6 JS engine variable can be represented with var, let and const. this was to overcomes var disadvantages.
* There is no need to put semicolon at the end of each line unlike in Java, but as a best practice it is better to add.
* 'typeof' command will show us what the datatype if a specific variable is even if we declare it as var, let or const.

Machine generated alternative text:
let a=4
console. log(a)
console. log(typeof(a))
let b = 234.6
console. log(typeof(b))

* Common JS datatypes are - number, string, boolean, null and undefined.
* Assignment can be done as let c= a+b
* One diff between var and let - we can't declare let twice with the same variable name but with var we can do it.
* Machine generated alternative text:
  vail c = “Rahul Shetty”
  console. log(typeof(c))
  let required = true
  console. log(typeof (required))
  I/null and undefined
  varc=a÷b
  console. log(c)
* But we can use let once as let c ="rahul" and latter as c=a+b it works. Only redeclaring doesn’t work with let, reassigning works perfectly.
* With boolean datatype we can use the NOT operator !
* Machine generated alternative text:
  let required = true
  console. log(typeof (required))
  I/null and undefined
  var c = a+b
  console. log(c)
  I/we cannot redeclare variable with let
  console. log( ! required)
* Const has blocked scope, we can neither reassign not redeclare
* For const, we can't change the value, but we can use NOT operator. Because here only the expression changes and not the value.
* if condition , while and do while loop works similar to how it is in Java
* Realtime usage of while loop in test automation is when we have to retrieve first n number of data from a table.
* Do while loop must end with a semicolon
* Whenever we know how much a loop should run beforehand itself that’s when for loop must be considered.
* Different for loops in JS - basic for loop, for of loop(similar to for each, mainly to iterate arrays), for in loop(mainly to iterate an object).

* Arrays in Javascript - below image shows some of the basic array operations. Push cmd to insert new elements at the end, Pop cmd to remove the last element, unshift cmd to insert a new elements at the beginning. Shift cmd to remove element from the beginning.
* Machine generated alternative text:
  1 var marks Array(6)
  2 var marks — new Array(2e,40,35,12,37,100)
  3
  4 var marks =[20,40,35,12,37,lOø]
  5 console.lcg(marks[2J) ¡/35 ___________________
  7 console.log(marks) //(20,4e,35,14,37,lOOJ
  6 marks[3J - 14 )
  8 console.lcg(marks.length) ¡/6
  9 .arks.push(65)
  le console.log(marks) //[20,40,35,14
  11 marks.() ðI,4e,35,l4,37
  37,l%,65J
  12 marks.unshift(12)
  13 console. log(.arks)
* Arrays in Javascript - slice method is used to split the array into another array, indexOf method is used to get the index location of a value in the array, includes method is used to check if a specific value is present in the array, if not then it would return false.
* Machine generated alternative text:
  var arks =(20,40,35,12,37,leOj
  subeiarks =.iarks.s1ice(2,)
  console. lo€(subMarks)

* Machine generated alternative text:
  console. lo€(.arks. indexof(lee))
  ir. the array
  console. log(.arks. includes(12e))
* Iterating an array, in the below image we can see iterating the array with the help of normal for loop and summing up the total count. Another alternative to this is usage of **reduce** method- which takes 4 parameters in total, sum being the first parameter and its default value as 0 which is the fourth parameter. Mark which is the second parameter resembles the marks[i] in the for loop and the third parameter sum+mark which is the operation that need to be performed which in this case is the summing up of numbers in the array.
* Machine generated alternative text:
  
* Iterating an array, in the below image we can see iterating an array with the help of normal for loop, where we are adding condition to check if number in array is even and then adding it to a new array. The same can be done with the help of **filter** method. Filter method takes 2 parameters one in the below example is score which is similar to scores[i] and the second parameter is the condition to be checked and would return a new Array by default.
* Machine generated alternative text:
  I/create new array with even numbers of scores array [12,14,16]
  var evenScores =[]
  for(let i =ø;icscores.length;i÷+)
  I if(scores[i] %2 == 0)
  {
  evenScores . push(scores[i])
  )
  console. log(evenScores)
  let newFilterEvenScores =scores.filter(score=>score%2==ø)
  console. log{(1newFilterEvenScores5j
* If the values in an array needs to be altered after performing some operation then we can use the map function
* Machine generated alternative text:
  let newFilterEvenScores =scores.filter(score=>score%2==ø)
  console.log(newFilterEvenScores) //[ 12, 14, 16 ]=>[36,42,48]
  I/map .
  let mappedArray=newFi1terEvenScores.map(score=>score3)
  console.log(mappedArray) ___ I
* We can also chain these 3 methods (reduce, filter and map) if we have multiple tasks to be performed on an array
* Machine generated alternative text:
  let sumValue4coresl.filter(score=>score%2==ø).map(score>score*3). reduce((sum,va1)z>sum÷val,ø)
* Sorting String array using sort method
* Machine generated alternative text:
  let fruits =[“banana’,”mango”, “pomegrante”, “apple”]
  fruits. sort()
  console. log(fruits)
* Sorting string array in reverse order
* Machine generated alternative text:
  let fruits =[ “banana”, “mango”,” pomegrante”, “apple”]
  console. log(fruits . sorte)
  conso1e.log(fruits.reversej)

**Scope of var, let and const**

* Var scope is at global level/ function level. If we declare and assign value to var inside a function then we won't be able to access it outside the function.
* If the var is declared global and also there is an if condition with same var variable outside function following the first var, then the value of second var will get printed at the end which is AFTERNOON. This means var is not having block scope, it has global scope only.
* Machine generated alternative text:
  I/var - global level/functional
  var greet = “Evening”
  if( 1 ==1)
  var greet = “Afternoon”
  function add(a,b)
  var hreet = “Morning’
  return a+b
  let sum =add(2,3)
  console. log(sum)
  console. log(greet)
* Now in the above example let us replace var greet with let greet, then instead of printing AFTERNOON it would print EVENING. Let is having block scope and hence it avoids confusion thus widely used than var
* Const has same scope as let, only change is it can't be reinitialized or reassigned. Hence it is widely used to store locator values in test automation space.
* Default value of var and let if not initialized is undefined.
* Const during declaration its mandatory to initialize a value to the same.

**Functions in Javascript**

* Are used to increase reusability of code, different ways of writing a function are - Function declaration, Function expression and arrow functions
* Function declaration
* Machine generated alternative text:
  function add(a,b)
  return a+b
  let sum =add(2,3)
  console. log(sum)
* Function expression - here the function doesn't have a name and we can directly assign it to a variable. (note that the below screenshot misses the passing of values)
* Machine generated alternative text:
  
* Arrow functions - makes function look even more simple
* Machine generated alternative text:
  let sumOfNumbers= (c,d»> c+d
  console. log(sumofNumbers(2, 3))

**Strings in Javascript**

* We can write string in single quote or double quote in JS.
* Finding the length of a String
* Machine generated alternative text:
  let day = ‘tuesday ‘
  console. logðay. length
* Making a substring of the above example using slice method
* Machine generated alternative text:
  let subDay = day.slice(ø,4)
  console. log(subDay)
* Fetching a character at a specific index
* Machine generated alternative text:
  console. log(day[1]) /lu
  —
* Using indexof method and counting how many occurances of a specific value is present in a String
* Machine generated alternative text:
  let newQuote =day+ “is Funday day
  console. log(newQuote)
  let val =newQuote. indexOf(’day”,5)
  console. log(val)
  I/tuesday is Funday
  let count = e
  let value =newQuote. indexOf( “day”)
  while(valuel== -1)
  count++ //2
  value =newQuote. indexof(”day” ,value+1)
  console. log(count)
* Split method to split the string into 2 halves
* Machine generated alternative text:
  let splitDay =day.split(”s”)
  console.log(splitDay[1] .length)
  console.log(splitDay[1] .trimQ.length)
* Parsing a string to integer(parseInt) and int to String(toString())
* Machine generated alternative text:
  let date = 23a
  let nextDate = ‘27’
  let diff = parselnt(nextDate) - parselnt(date)
  console. log(diff)
  diff.toString ____
* Concat using + operator
* Machine generated alternative text:
  let newQuote =day+ “is Funday”

**Javascript Objects**

* Object is a collection of properties, object can be created with the help of curly braces, the properties can be created as key value pair. And we can access the value of the properties with the key.
* Machine generated alternative text:
  let person = (
  firstName: ‘Tim’,
  lastName : ‘)oe’
  console. log(person. lastName)
* We can also access the properties using array like notation
* Machine generated alternative text:
  console. log(person[ ‘lastNam&])
* We can also update the object properties, note that we declared person of type let which allows us to reassign values.
* Machine generated alternative text:
  person.firstName =‘Tim Dane’
  console. log(person.firstName)
* We can also add new properties to the object
* Machine generated alternative text:
  person.gender = ‘male’
  console. log(persor)

* Machine generated alternative text:
  { f irstt4e: Tim Dane , 1astNe: 3oe, gender: Ìa1e )
* We can also delete a property using the delete command
* Machine generated alternative text:
  delete person.gender
* In keyword can be used to check if a property is present in the object or not
* Machine generated alternative text:
  conso].e.log(’gender’ in person)
* Iterating the object with the help of for loop
* Machine generated alternative text:
  I/print all the values of the javascript objectj
  for(let key in person)
  {
  console.log(person[key]) .
* We can add numbers and even functions to an object in Javascript
* While calling Strings or Number properties in an object we can call them as object.key but while calling a function we need to call them as object.function() with the parenthesis.
* Machine generated alternative text:
  let person = (
  firstName: ‘Tim’,
  lastName : ‘Joe’,
  age:24, Ej
  fuliName : function()
  console. log( this . firstName+this. lastName)
  }
  console. log(person . fuflNameQ)

**Javascript Classes**

* Classes were introduced in ES6 engine, before that everything was wrapped in an object.
* We can use getter methods(get) to define the property. While calling the property defined using getter method we can call them outside the call without using parenthesis. Do note that Get properties and functions are different.
* We can create constructor inside the class, which gets executed before everything else in a class. For this we can use this keyword to assign the variable to be accessible inside the whole class, if we don't then the parameterized values would be available only inside the constructor.
* While using parameterized constructor we can assign the value during runtime and more importantly as shown in the below screenshot, we can create different objects and create different instances, hence the variables inside the constructor is known as instance variables.

* Machine generated alternative text:
  class Person
  age=[25
  /1 location =“canada”
  get location()
  {
  return “canada”
  )
  //constructor is method which executes by default when
  con structor(firstName, lastName)
  this.firstName =firstName
  this.lastName = lastName
  )
  I/methods ________________
  fullName()
  {
  console. log(this . firstName+this. lastName)
  1
  - JI
  let person =new PersonÇTim’,”)oseph)
  let personi =new Person(”CHris,”Ones”)
  console. log(person.age)
  console. log(person . location)
  console. log(person . fullNa.eO)
  console. 1og(person1.ful1Name)
* We can also export a class and use its features in another js file. For this we have to do first export and then import, export is achieved with the help of the command module.export
* Machine generated alternative text:
  Seis ) uwner? J oasILJs t <unicnown> .‘ q i’erson
  . inodule.exports = class Persor
  {
  age=25
  II location =“canada”
  get location()
* Once exported we can use the require command to import the class into another JS file and then we are able to create objects of the class in the new file. For automation perspective this helps us in creating classes with page objects and action methods and reusing them in Test class.
* Machine generated alternative text:
  ‘‚
  J Welcome 35 basics5.js X 35 bascs6js 35 basics7gs •
  c>Usecs>Owner> is basics5.js>CeI Person
  1 const ii’ãriI = require(’./basics7’)
  2 let day = ‘tuesday
  3 console.log(day.length) I/B
  4 let subDay = day.slice(ø,4)
  5 console.log(subDay)
  6 console.log(day[1]) f/u
  7 I/tue day
  ÁÇJJ________

**Inheritance in Javascript**

* Implemented using extends keyword like in Java
* Along with that we need to have parent class exported and imported in the subclass, also if the parent class has a constructor then child class should have the same constructor with the first line as super() method to call the parent class constructor(if it is a parameterized constructor then parameters should be passed in super)
* Machine generated alternative text:
  const Person = require(”./basics7”)
  class Pet extends Person
  {
  coñstructor(firstName, laštÑamë)
  {
  I/call parent class constructor
  superfirstName,lastNameÍj1 ____________________
  _iz 1
* Once the above is done we can access all the methods and properties(getter methods) in the subclass directly. We can create an object of the subclass and call the methods in parent class.
* If both the parent class and child class has functions or getter methods with the same name then while invoking the same the parent class method will be overridden and the subclass method will get executed.