

1.

"Everything in Javascript happens inside an Execution

Exection content

A STATE OF THE PARTY OF THE PAR	
Memory (var. environ)	Code (Thread of exam)
key : Value .	0
a : 10	0
fn: \\ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \	O
•	0
Har go	201 8 9 9 1 1 2 1
(600)	i side appli

Javascript is a sychronous single-threaded language.

Specific order only execute one command

what happens when Javascript code is executed?

= Execution Content is created

Vor n = ?;

Function square (num) {

var oru = num * num;

return anu;

}

var square2 = square(n);

Von SquareH = square (4);

Memory code.

n: undefined Memory code:

square: {...}

square: {...}

square: {...}

square: Undefined Memory return

square: Undefined Memory return

square: Undefined Memory return

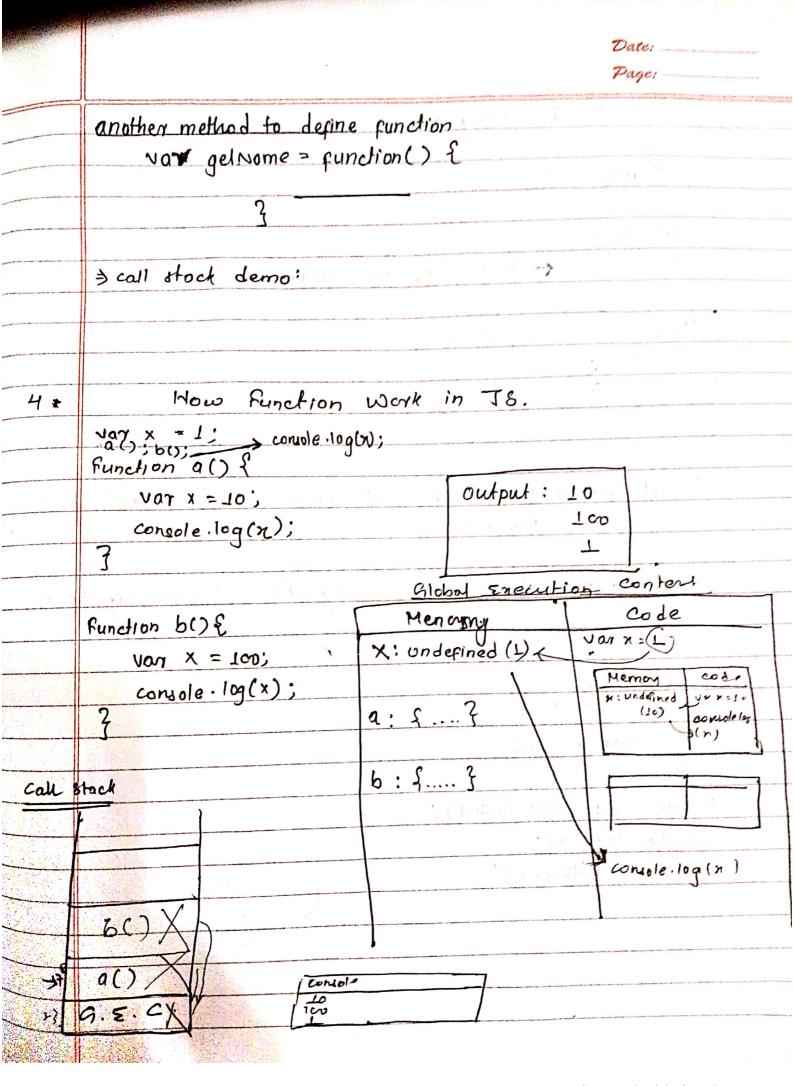
square: Undefined Memory return

square: Texton

and Texton

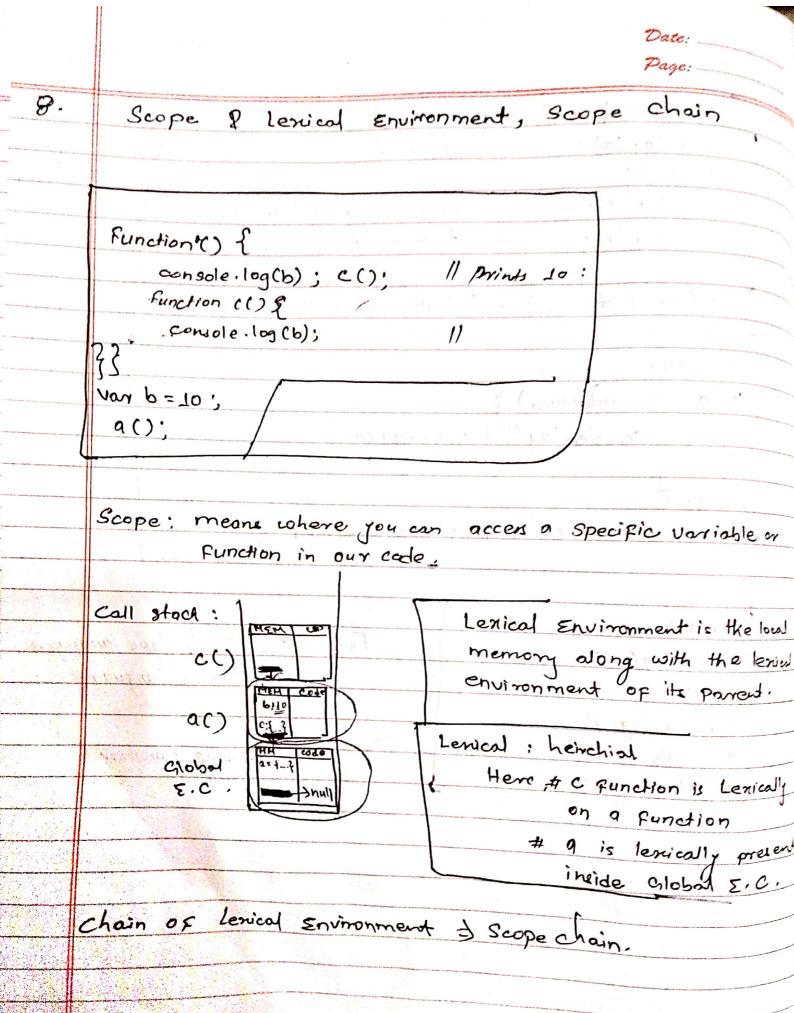
{ 1 th Phase: Memory allocally creation. }

"Call stock maintaine the order of execution content." Call stock - Execution content stock. - Program Stack - Control Stock. - Runtime Stock. - Machine Stock. - Machine Stock. - Machine Stock. - Warriables & Function. # console log() * Prints on consolo. # Arrow function. Vor getName = () > {		
Call stock - Execution Content stock. - Program Stack - Control Stock. - Runtime Stock. - Machine Stock. - Warring in Javascript (Variables & Function). - Warring: access anywhere below or apter, - Console log() & Prints on condo - Arrow Function: Vor getName = () > {		execution content "
Call stock - Execution Content stock. - Program Stack - Control Stock. - Runtime Stock. - Machine Stock. - Warring in Javascript (Variables & Function). - Warring: access anywhere below or apter, - Console log() & Prints on condo - Arrow Function: Vor getName = () > {		"Call stack maintoine the order of
- Execution Content stock. - Program Stack - Control Stock. - Runtime Stock. - Runtime Stock. - Machine Stock. - Warriables & Functiony). Worting: access onywhere below or apter. # console log() & prints on consolo # Arrow function: Vor getnome = () >> {		
- Execution Content stock. - Program Stack - Control Stock. - Runtime Stock. - Runtime Stock. - Machine Stock. - Warriables & Functiony). Worting: access onywhere below or apter. # console log() & prints on consolo # Arrow function: Vor getnome = () >> {		Call stack.
- Program Stack - Control Stock Runtime Stack Machine Stack Machine Stack. - Machine Stack.	-	contin content stock.
- Runtime Stack, - Machine Stack. Hosting in Javascript (Variables & Functions). Worting: access anywhere below or arter, # console log() & prints on consolo: # Arrow function: Vor getName = () => {	-	- Or stand
- Runtime Stack, - Machine Stack. Hosting in Javascript (Variables & Functions). Worting: access anywhere below or arter, # console log() & prints on consolo: # Arrow function: Vor getName = () => {	-	- Program - All Control of the Contr
Horting in Javascript (Variables & Functions). Hosting: access anywhere below or apter. # console log() & prints on consolo. # Arrow function: vor getwome = () >> {	-	Control Stock
Horting in Javascript (Variables & Functions). Hosting: access anywhere below or apter. # console log() & prints on consolo. # Arrow function: vor getwome = () >> {		- Runtime Stack.
Horting in Javaseript (Variables & Function). Wosting: access onywhere below or apter, # console log() & prints on consolo; * Arrow function: Vor getName = () > {	A CANADA	- Machine Stack.
Hosting: access anywhere below or apter, # console.log() & prints on consolo: * Arrow function: vor getName = () => {	Section Street	
Hosting: access anywhere below or apter, # console log() & prints on consolo: * Arrow function: vor getName = () => {		
Hosting: access anywhere below or apter, # console log() & prints on consolo: * Arrow function: vor getName = () => {		
Hosting: access anywhere below or apter, # console.log() & prints on consolo: * Arrow function: vor getName = () => {		
Hosting: access anywhere below or apter, # console.log() & prints on consolo: * Arrow function: vor getName = () => {		11 1. A seriot
Hosting: access anywhere below or apter, # console.log() & prints on consolo: * Arrow function: vor getName = () => {		Horting in Javascripi
Hosting: access anywhere below or apter, # console.log() & prints on consolo: * Arrow function: vor getName = () => {		(Variables & functions).
# console.log() & prints on consdo. * Arrow function. vor getnome = () => {		
# console.log() & prints on consdo. * Arrow function. vor getnome = () => {		blad: - access onywhere below or after,
# console.log() & prints on consdo. * Arrow function. vor getnome = () => {	#	Mosting: account
Arrow function: Vor getName = () => { conside. log ("Namarte Javascrapt") } Function. Function get Name() { conso.log ("Namarte Javascript") }	4	
Arrow function: Vor getName = () => {		# console.log() . & Prims on all
console. log ("Namarte Javascript") Function. Function get Name() { conso.log ("Namarte Javascript")		BEN THE STREET STREETS IN THE STREET
console. log ("Namarte Javascript") Function. Function get Name() { conso.log ("Namarte Javascript")	#	
console. log ("Namarte Javascript") Function. Function get Name() { conso.log ("Namarte Javascript")	+	a ligh.
Console. log ("Namarte Javascript") Function. Function get Nome() { conso.log ("Namarte Javascript") ?	*	Arrow functions
Function. Function get Nome() { conso.log ("Namarte Tawascript") ?		voi gethame = () => {
Function. Function get Nome() { conso.log("Nemarte Tawascript") 7.		console. 109 ("Namarte Javascript")
Function. Function get Nome() { conso.log("Namarte Javascript") ?		
function get Nome() { conso.log ("Nemarte Tawascript") 7.		The first warm of the first water
function get Nome() { conso.log ("Nemarte Tawascript") 7.		The state of the s
conso.log (Namarre Javascript")		Function.
conso.log (Namarre Javascript")		function get Nome() 7
The end of the second of the s	1	conso.log ("Namarte Tawascript")



	Date:	
	Page: —	
6.	Undefined and Not defined	
	console. log (a); Il undefined	
	Var a = 7;	1
	console. log (a); Il Defind and print as 7	
	consde. log(2); Il hot defined.	
	The state of the s	
	underined # Empty = but as called as placeholder.	
	Street 4 21/p) = 31 25	
	a = 10.	
	If (a === undefined) {	and "
	console. log ("a is underined")	p
	7	
	else {	
n n	console.log ("a is not do undefined");	5- 1-
	Short was at mother to	
L712	Var a-; and the last to the sets	Herb
	Township / long /	e language
	console log(a);	24000

	else {	
	console.log (a is not	de underined ");
		size the mi netternit
_	Var a ; I was tasks !	· Abele New
1	console log(a);	Javascript: lossely type language
	0=10;	Javascript: lossel, type language weally language
-	console log(a);	< J
	9="hello world"	Variable can store anything
	convole. log (a);	ary tury
	Haritain to the	
	VALUE OF THE STATE	
-	(15/10 15/15)	
1		
- more		to surveying fusher 30 mining
1		
1		



	Date:
	Page:
1.	Temporal Dead zone Let & const in Ts.
NAME OF TAXABLE PARTY.	
	"Let & contres declaration are Hoisted".
	console.log(b); 11 undefined.
	Var a = 10;
lgel k arak NV Patrasad	vor b = 100;
	cornole.log(a); Il error. connot acces before.
- Marie Contraction of the Contr	let a = 10; w Reference error.
o management of the second	vor b = 1cv;
	let a =10;
	console-log(a); Il prints 10.
	vo7 b = 10v;
	Marian Company of the
	Control of the contro
	"- I dead zone" is the time since then when let variable
	is hoisted and till it is initialize some value. The time
	betwon them is temporal dead zone.
******	let a = 10; // Syntox error.
	let a = las:
	var 10 = 10; // No error.
	Nor b = 1 co ;
Processia and	

	Date:
	Page:
	Let a =10; } separate memons space.
	Cont b = 100;
	Textions of prochambels of the
	let a;
	const b = 1000; fine and prints 10.
	a = 10;
	convole·log(a);
	Leto;
	const b; / Synton evror (missing initialization);
	b = 100; ~ // missing initialization in const declarate
	a = 10;
	10° 10°
	The state of the s
44-3-	const is more strict than let".
	Const should be initialized and declared together.
	Let and const don't allow dyplicate declaration "
is es	tell make my dear out out out it " and hook house
Star Co	1st place or constant: Use const 1
	: Use Let @
	e later var 3.
	Avoid temporal dead zone ine move initializations error,
	The error,

Date:	
Page:	

let a = 1000;

conf. b = 1000;

b = 1000;

// Type error, (Assignment to comfant variable)

a = 10;

let a = 100;

// Syntan error.

Reperence ermon: When program tries to kind a specific variable inside the memory space and you cannot access it

Console. log (a); (Reference error) or when not defined let a = 1900;

Three ways to declare a variable: const (1)

Let (2)

Var. (3)