# Dag 5

about:black

console.log(window)

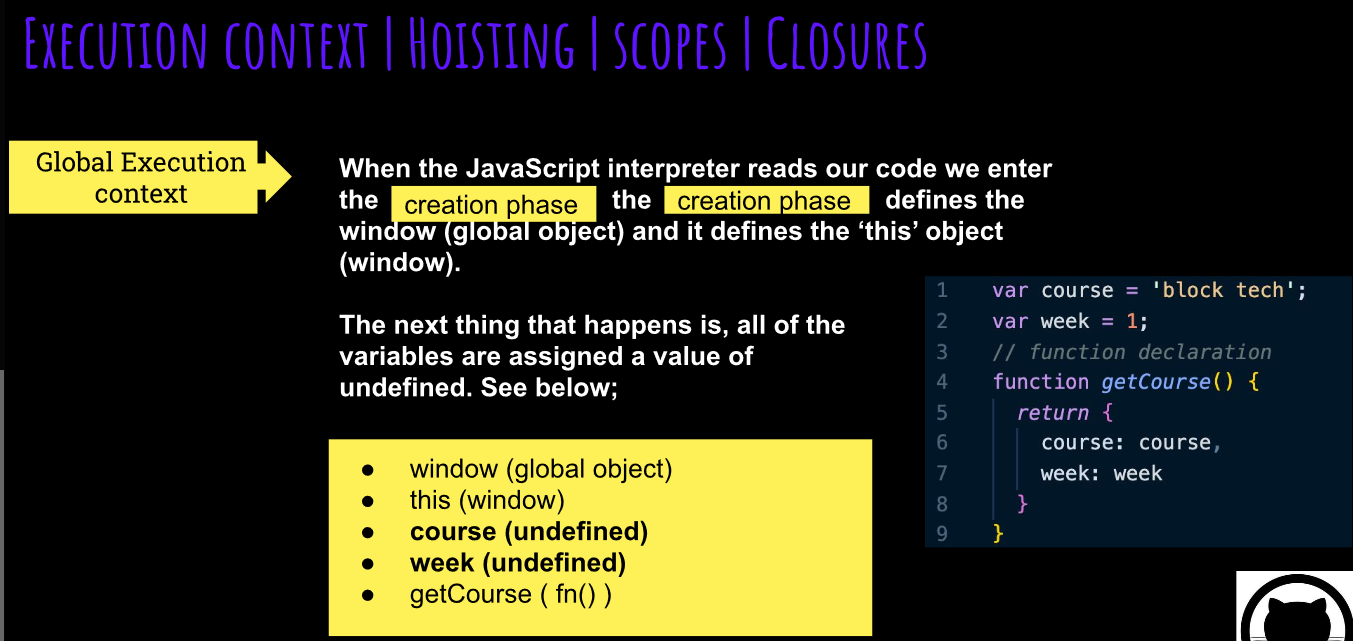
window.innerHeight

## Global execution context

window (equals to global object) = object waar alle browsereigenschappen in zitten

this (equals window)

global (equals global object)

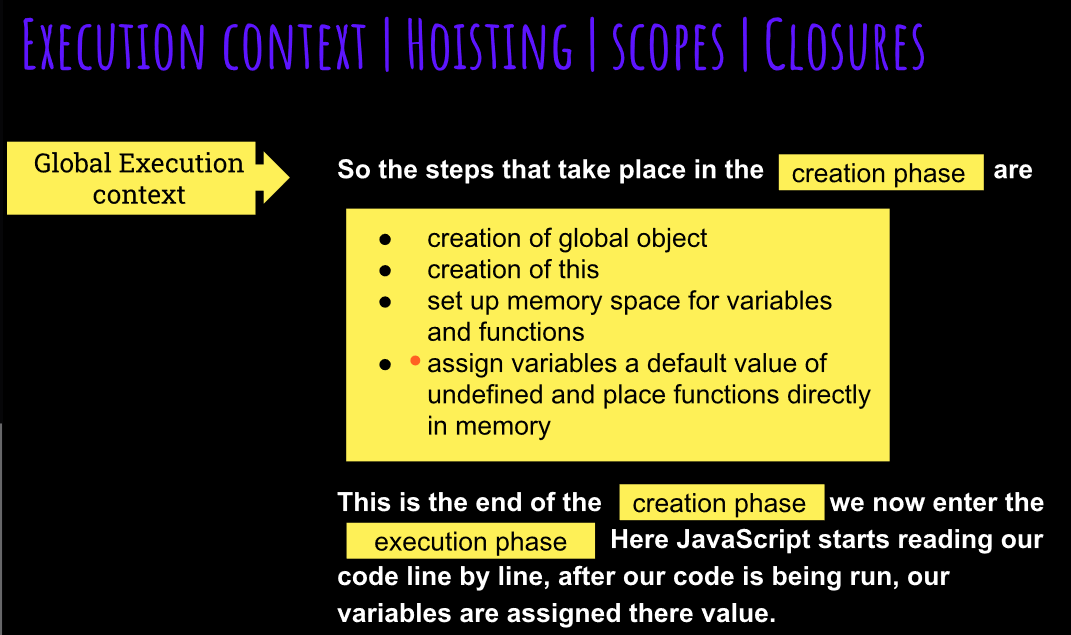


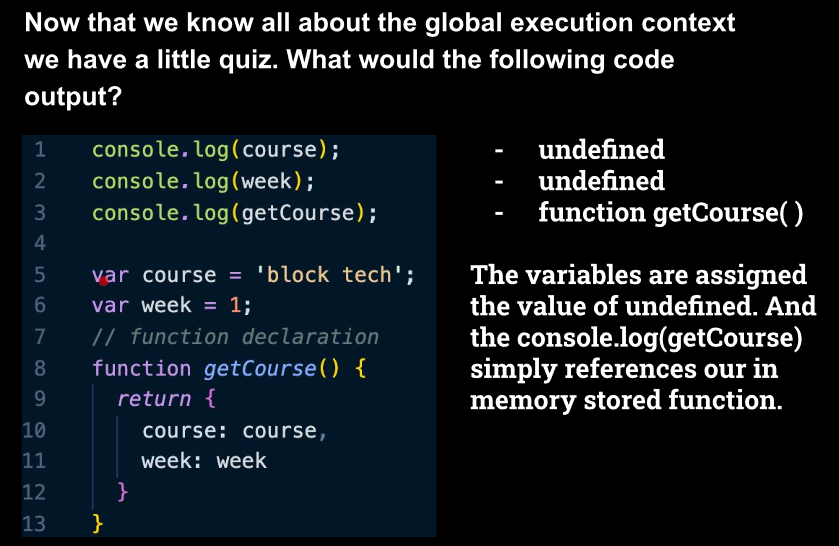
declaratie = variabele aanmaken

assignment = vullen van variabele

In creation phase worden alleen de declaraties van variabelen en functions gelezen (gehoist: naar boven gehaald), niet de assignments. Zo wordt er door JS ruimte gealloceerd voor de variabelen en functies.

De variabelen worden gevuld in de execution phase.





want??

**function:**

maakt een arguments object ipv een global object

**scope**

de huidige context van de execution

**hoisting**

variabelen en function declarations fysiek verschuiven naar de bovenkant van de code (onderdeel van creation phase)

console.log(num);

var num;

num = 6;

* num = undefined, want var wordt gehoist, daarna console.log en daarna num vullen

**closure**

zorgt ervoor dat de inner function toegang heeft tot de outer function, ook als de execution context van de outer function al is gebeurd

(functie binnen functie (outer & inner function))

**global scope vs local scope**

global scope: declared once accessible anywhere

local scope: alleen binnen het block waarin t gedeclareerd is

**function scope vs block scope**

function scope: variables inside the function

block scope: een bepaald block is relevant voor die conditie, bijv. if of for loop statements

**let** mag je niet opnieuw declareren, maar **var** wel