Verkefni 2   
**(7%) 1. Hvað er ECMAScript?**   
ECMAscript er svona viðurkenndur staðall fyrir scriptu tungumál og er viðurkennt í ECMA-262 og ISO/IEC 16262. Það byggist á javascript og er mest notað fyrir skriptur og vefnum og þá mest client side , en það eru að aukast notkunin fyrir server forrit og þjónustur með að nota node.js

**2. Hvað þýðir að JavasScript sé loosly typed tungumál?**

Þú þarft t.d. ekki að gera int x = 10 eða string nafn = „name“, þú definear bara breytu með var x = 10 og þá er það int, eða true/false eða með streng, og vélin svona sjálfkrafa veit og gerir ráð fyrir hvað þú vilt og eins ef þú gerir villur reynir að skilja þig samt og keyra. það meirasegja keyrir bara x = 10 með að sleppa var nema þú sért í strict mode

3. Er einhver munur á einföldum og tvöföldum gæsalöppum í JavaScript?   
nei bæði marka streng í javascript hinsvegar þegar er komið að json eða öðru er betra að venja sig á “

**4. Hvað er null og undefined?**

Þegar breyta er null þá er hún jafnt og null semsagt hefur það gildi hinsvegar ef hún er undefined þá á eftir að gefa henni gildi þá er hún hvorki false eða true eða strengur eða tala

**5. Hver er munurinn á == og === í JavaScript?**

== er jafntog === er það sama

t.d.

var k = 1;

var u = true;

if(k == u) /þetta er true, false = 0 true = 1 þannig þessi setning gerist þannig að **true er jafntog 1**

if(k === u) /þetta er false, til þess að þetta væri true þyrfti k að vera líka booelan breyta með true

=== verður að vera alveg eins eða identical ekki dugar að skila sömu útkomu heldur að verður að vera að sama formi, int, bool, string etc..

sama dæmi er hægt að gera svona

var k = „ten“

var u = 10

k == u er true

k === u er false

**6. Afhverju virkar eftirfarandi kóðabútur með óskilgreindu breytunni, útskýrðu?**   
var result = (true || someUndeclaredVariable); alert(result); // true

|| er or þannig að result er = true eða einhverónefnd breyta, og þar sem það e renginn ónefnd breyta verður það frekar true

**7. Hvað er regular expression? komdu með dæmi!**

Það er röð af stöfum sem mynda leitarmynstur, þegar það er leitaf að gögnum í texta getur þú notað þetta leitarmynstur til að lýsa hvað þú ert að leita af, **Regular Expression** getur verið einn stafur eða flókara, og hægt er að nota það til að framkvæma allskonar textaleit eða jafnvel skipta út texta

var mynst = /tskoli/i

hérna er /tskoli/ regular expression og þá tskoli leitarmynstrið og lila i segir að þetta er case-insensitive semsagt bæði hástafir og lágstafir koma til grein**a**

**7. Hvað er expression? komdu með dæmi! //Uppfært**Expression í Javascript er það sem býr til gildi og þar sem er búist við gildi, nema þegar þú ert kominn í aðgerð eða einhverskonar loopu, eins og t.d það sem flokkast undir Expression í javascript er var a = 3, þarnea er gildið 3 komið í a, annað dæmi er t.d. var a = 3 + 8, þetta er kallað expression,   
  
Statement væri síðan ef ég væri með function sem tæki inn 2 breytur og myndi leggj aþær saman, t.d.  
function leggjasam(a,b)  
{  
return a+b;  
}Þetta er flokkað sem statement meðan hitt er expression

**8. Skilgreindu fall á þrjá mismunandi vegu með kóðadæmi.**

function fall1(){

return ("þetta er eitt fall");

}  
  
var fall2 = function(){return ("annað fall");};

(function fall3(hallo){

return hallo + ‘ heimur‘;})  
(„halló“)

**9. Hvað gerir let?**   
let svosem virkar svipað eða eins og var nema hvað varðar scopeið eða hvar þetta gildir, let er meira svona local breyta og tilheyrir bara því scope það er eða þangað til það er lokað}, meðan var tilheyrir bara næsta functioni sem það er, en ef annað þeirra er skilgreint utan function eða slaufusviga þá eru þær taldar global, let t.d. er gott að nota í for lykkjum því þá eftir að for lykkjan er lokuð þá er let breytan bara dottin úr gildi en hinsvegar myndi var breytan vera enn í gildi meðan functionið er ekki lokað.

**10. Útskýrðu Scoping í JavaScript.**Scoping er svona hvaða svæði viðeigandi fall eða breyta tilheyrir, t.d. var breytur sem eru ekki hluti af neinu functioni fara bara efst og eru kallaðar global breytur og allt sem er í scriptunni hvort sem það er function eða bara skipun geta kallað í þessar global breytir því allt sem er undir global scopeinu hefur aðgang að global breytunum, hinsvegar ef ég er með function og það er með var og kannski annað fall sem er líka með var, þá raðast það bara í toppinn í því functioni ég get notað þá breytu með global breytu inní fallinu en ég get ekki látið síðan annað fall sem er fyrir utan þess fall eða child fallið nýta sér local breytuna í fallinu, því það er fyrir utan þess svæði, en það fall gæti nýtt sér global breytuna...

**11. Hvað er hoisting í JavaScript, komdu með dæmi?**

Javascript engineið endurraðar kóðanum okkar í keyrslu eða semsagt allar breytur og function sem tilheyra ekki neinu sérstöku falli eða block kóða er fært efst og allar þær sem tilheyra öðru falli er fært efst í því falli og svo framvegis þannig að það skiptir ekki máli hvort þú kallar í functionið áður en þú býrð það til því það verður raðað.

t.d.

leggjasaman(10,10); ,myndi skila 20 því að þrátt fyrir að ég kalla í þetta á undan þá er hitt í raun keyrt á undan,

function leggjasaman(a,b){

return a+b;

}

Sama á við þetta

Leggjasaman(k,z) #4

var k = 20; #2

function leggjasaman(a,b){ #3

return a+b;

}

var z = 30; #1

**12. Hvað gerir 'use strict' í JavaScript kóða?**

Use strict er einsog það segir sjálft gerir þetta strangara, ekki sama loose tungumálið, ef þú ert með use strict er ýmislegt sem þú færð ekki að gera t.d.

* Gera x=10, þú verður að define-a breytuna með var x fyrst.
* Eins með array eða aðra objecta verður að defina þá fyrst.
* Getur ekki delete-að breytu eða objectum.
* Deleta föllum.
* Passar að þú sért ekki að nota sama breytuheitið tvisvar í sama scopei.
* Getur ekki notað escape charactera.
* Getur ekki sett inn octal numeric literals.
* Getur ekki notað with skipunina.
* Og í föllum þegar maður gerir this.xx var global object verður undefined.

Það verður samt að nota use strict í byrjun á skriptu eða efst í falli til að nota það.

**13. Hér er dæmi um “function decleration” og kall á fallið. Afhverju virkar eftirfarandi kóði í JavaScript, útskýrðu? x(); function x() { console.log('x'); }**

Þetta fórum við áðan í gegn með hoisting þarna er byrjað að kalla í fallið x() og í fallinu x er verið að logga í consoleinn strenginn x og eins og þessu er raðað upp þá það sem gerist fyrst er að fallið er lesið og svo kallið í fallið þannig að þetta skilar x

**14. Í hvaða röð er kóðinn keyrður í raun eftir að þýðandinn (e. interpreter) er búinn að fá hann til sín? Hvað birtir alert(), afhverju?**

**function foo(){ function bar() { return 3; } return bar(); function bar() { return 8; } } alert(foo());**

Fyrst erum við með foo sem býr til local functionið bar í sér sem skilar 3 og þegar það er búið skilar foo útkomunni úr bar sem er 3 en síðan keyrist fallið bar sem er global function og það yfirskrifar barið sem kom úr foo þannig að niðurstaðarn verður að það verður returnað 8.

**15. Hvað gerist þegar ég gleymi að skilgreina breytu með var? Útskýrðu hvað gerist age = '30'; console.log(age);**Þar sem javascript er loose þá reddar það málunum fyrir þig í þessu tilviki og gerir ráð fyrir að þú ætlaðir þér að gera streng þannig að þetta birtir bara age breytuna í consoleinn eins og þú hefðir gert var age = ‘30‘

**16. Útskýrðu hvað eftirfarandi kóði gerir, hvað gera svigarnir? (function() { alert('Hello World'); })();**Þegar þessu er háttað svona þá er fallið að kalla í sig sjálft strax eftir og það er gert þannig í þeirri línu sem þetta er sett þá fer það í gang, þannig funcionið virkar eins og hvert annað function nema svigarnir keyra það í gang strax.

**17. Hvað er Template literals? komdu með dæmi!**Breytir því hvernig er að vinna með strengi og breytur eða escape charactera, þetta er nýtt í ECMA6 og virkar þannig að í staðinn fyrir að þurfa vera með t.d.  
console.log(„Halló „+args+“ góðan dag“) er hægt að gera  
console.log(`Halló ${argns} góðan dag`) eða í stað fyrir  
console.log("Hér er ein flott lína \n" +  
"og hér er önnur"); er hægt að gera  
console.log(`Hér er ein lína  
og hér er önnur`);

þannig að það getur mögulega verið þægilegra að vinna með langa flókna strengi í staðinn fyrir að þurfa vera að plúsa hitt og þetta við

**18. Hvað er call stack og hvað er heap?**Call stackið er í raun hvernig kóðinn er raðaður upp af engine-innu í hvaða röð líka, virkar svolítið eins og vinnsluminni, eins og þegar þú keyrir kóðann upp þá er hann kominn fyrir í call stackinu og global breyturnar fyrst og síðan vinnur hann með það sem hann þarf.  
  
heap er síðan þar sem objectinn eru geymd til þess að þau séu fljótlega aðgengileg og eru þau endurröðuð þar í forgangi eftir því, hinsvegar efmaður er með rosalegt verkefni þá vill maður hafa scriptuna sem fljótlegasta og ef það er eitthver breyta eða object sem maður veit maður mun 100% ekki nota aftur eftir að hún er keyrð getur það létt á heapnum að geta delete skipunina.